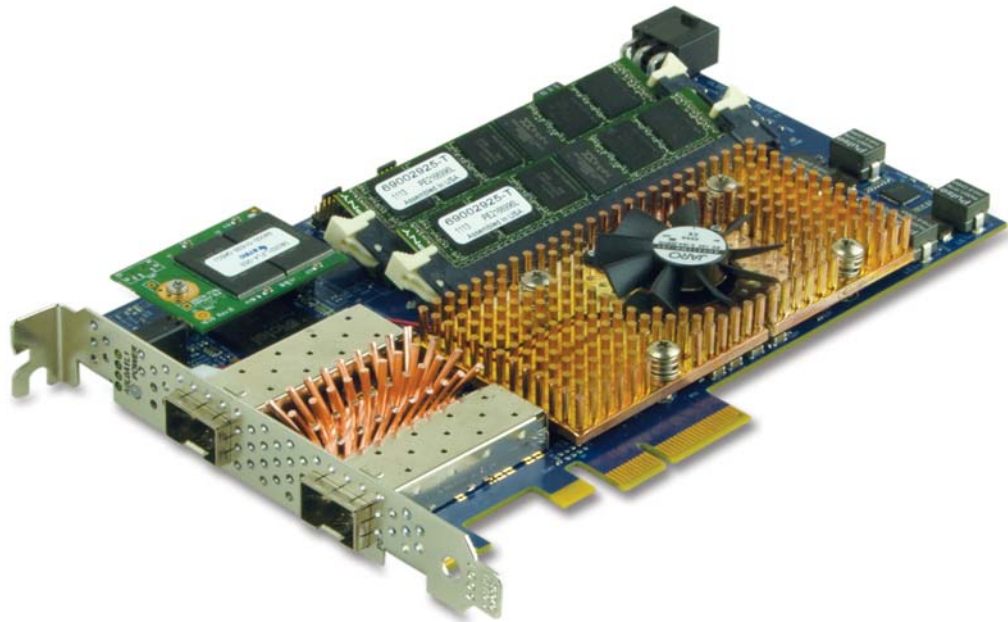


Reference Manual

WANic*-66512 Packet Processor PCI Express 10-Gigabit Ethernet First Edition

Part No: 87003034-820



imagination at work

© 2012 GE Intelligent Platforms Embedded Systems, Inc. All rights reserved.

An asterisk (*) indicates a trademark of GE Intelligent Platforms, Inc. and/or its affiliates. All other trademarks are the property of their respective owners.

Confidential Information - This document contains Confidential/Proprietary Information of GE Intelligent Platforms, Inc. and/or its suppliers or vendors. Distribution or reproduction prohibited without permission.

THIS DOCUMENT AND ITS CONTENTS ARE PROVIDED "AS IS", WITH NO REPRESENTATIONS OR WARRANTIES OF ANY KIND, WHETHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO WARRANTIES OF DESIGN, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. ALL OTHER LIABILITY ARISING FROM RELIANCE UPON ANY INFORMATION CONTAINED HEREIN IS EXPRESSLY DISCLAIMED.

Table of Contents

Preface	11
1 • Getting Started	15
1.1 Handling Precautions.	16
1.1.1 Electrostatic Discharge (ESD) Precautions	16
1.1.2 Optical/Laser Precautions	17
1.1.3 Safety Precautions	18
1.2 Environmental Requirements.	19
1.3 Air Flow and Cooling Requirements.	20
1.4 Power.	21
1.5 Required Tools.	22
1.6 Installation Requirements.	22
1.7 Unpacking Procedures.	23
1.8 WANic-66512 Front Panel.	24
1.8.1 SFP+ Housing Bays	25
1.8.2 LEDs	26
1.9 Hardware Installation.	27
1.9.1 Installation Overview	27
1.10 SFP+ Modules Installation and Removal.	27
1.10.1 SFP+ Modules Installation	28
1.10.2 SFP+ Module Removal	29
1.11 WANic-66512 Installation and Removal.	30
1.11.1 WANic-66512 Installation	30
1.11.2 WANic-66512 Removal	32
1.12 Software Installation.	33
1.12.1 Additional Software	33
1.12.2 Operating System Support	33
1.12.3 Cavium Networks Users' Organization Web Site	34
2 • Hardware Description	35
2.1 Overview.	35
2.2 Features.	36
2.3 Multi-Core Processor.	37
2.3.1 HFA/DFA Engine	39
2.3.2 Boot Bus	39
2.3.3 QLM Packet Interfaces	40
2.3.4 Processor Interfaces	41
2.3.5 Watchdog Timers	45
2.3.6 Reference Clock	47
2.4 Synchronous Ethernet Clock (SyncE).	48

2.5 FPGA	49
2.5.1 Control and Status Registers	50
2.5.2 Persistent Memory Interface (PMEM)	50
2.5.3 FPGA Clocking	50
2.5.4 FPGA Interrupts	50
2.5.5 FPGA Reset Control	50
2.5.6 I2C Interface	51
2.5.7 TWSI Interface	51
2.5.8 FPGA Power Supply Control	51
2.5.9 On-Board Thermal Protection	52
2.6 Memory	53
2.6.1 Internal (Level 2) Cache Memory	53
2.6.2 DDR3 SDRAM	53
2.6.3 Boot Flash	54
2.6.4 Persistent Memory (PMEM)	54
2.6.5 Serial EEPROM	55
2.6.6 USB Flash Drive	55
2.7 Gigabit Ethernet Physical Device	56
2.8 Power Distribution	57
2.8.1 PCIe Edge Power	57
2.8.2 IR Subsystem	58
2.9 Temperature Management	59
2.9.1 Monitoring Temperature from the Host	59
2.10 Headers	60
2.10.1 J5 Header	61
2.10.2 J6 Header and JTAG Scan Chain	62
2.10.3 Enhanced JTAG Header (EJTAG)	63
2.11 Dual In-Line Package (DIP) Switch	64
3 • FPGA Registers	67
3.1 FPGA Register Summary	67
3.1.1 Temperature Sensor Interrupts	68
3.2 Register Descriptions	69
FPGA Revision Register (0x00)	70
Board ID and DIP Switch Register (0x01)	71
LED Control Register (0x02)	72
Interrupt Control Register (0x03)	73
Interrupt Status Register (0x04)	74
Peripheral Reset Register (0x05)	75
Transmit Control Register (0x06)	76
Fan and Temperature Status Register (0x07)	77
I2C Registers	78
I2C Control Registers (0x08, 0x0C, 0x1C, 0x18)	80
I2C Address Low Register / I2C Address High Register (0x0A, 0x0E, 0x1E, 0x1A / 0x09, 0x0D, 0x1D, 0x19)	81
I2C Data Registers (0x0B, 0x0F, 0x1F, 0x1B)	82
Boot Flash Upper Address Control Register (0x20)	83
Miscellaneous Functions Register (0x21)	84

3.2 Register Descriptions (continued)	
PHY Interrupt Register (0x22)	85
Thermal Interrupt Mask Register (0x23)	86
IR3541 Interrupt Register (0x24)	87
DIMM Thermal Event Register (0x25)	88
4 • Software Description	89
4.1 Software Overview.	89
4.2 Cavium Software Development Kit.	91
4.2.1 OCTEON Ethernet Driver	91
4.2.2 Cavium Embedded File System	92
4.2.3 Simple Executive Library	94
4.3 User Application.	94
4.4 U-Boot BootLoader.	95
4.4.1 Boot Process	95
4.4.2 Building U-Boot	97
4.4.3 U-Boot Commands	98
4.4.4 PCI Console Redirection – U-Boot	101
4.4.5 PCI Console Redirection – Linux	102
4.4.6 Environment Variables	103
4.4.7 U-Boot Scripting	104
4.4.8 DHCP	104
4.4.9 Automated UA Executable Load and Boot	106
4.4.10 EEPROM Mapping and Tuples	107
Checksum Validation	110
Load and Boot Information	111
Memory Testing	112
PHY Testing	113
Primary File Information	115
RLDRAM Testing	116
SerDes Testing	117
Script	118
Script Active	119
Source IP Address Information	120
Secondary File Information	121
UART Configuration	122
UART MMC Interface	123
U-Boot Normal Image Version	124
4.4.11 EEPROM Tuple Update and Enumeration	125
4.5 Linux Support Packages.	127
4.5.1 BSP	127
4.5.2 LSP	127
4.5.3 Flash Driver	129
4.5.4 NPA Driver	134
4.5.5 Linux In-Service Daemon	143
4.5.6 Diagnostics Application	143
4.6 Additional Features.	144

4.7 Examples.	145
4.7.1 Linux Applications Examples	145
4.7.2 Simple Exec Application Examples	147
5 • Linux Support Package (LSP) Applications	151
5.1 Diagnostic Utility.	151
5.1.1 Setup	153
5.1.2 Running the Diagnostic Utility	154
5.2 Main Menu Options.	156
View Test Results	156
EEPROM Menu	157
Run Auto Test	158
Auto Test Configuration Menu	159
Diagnostic LED Test	161
Flash Menu	162
Log File Location	163
Show Core Temperature	164
Memory Test	165
USB Storage Test	166
5.3 In-Service Daemon.	167
6 • Specification	169
6.1 Hardware Specifications.	169
6.2 Regulatory Compliance Notice.	170
6.2.1 Emission Notices	170
A • GE Intelligent Platforms, Inc. Software License Description	171
B • GNU General Public License V2	173

List of Figures

Figure 1-1 WANic-66512	20
Figure 1-2 WANic-66512 External Power Connector	21
Figure 1-3 WANic-66512 Front Panel	24
Figure 1-4 Inserting or Removing an SFP+ Module.	28
Figure 1-5 Installation in a PCI Bus Chassis	30
Figure 1-6 Connecting to the Power Supply	31
Figure 2-1 WANic-66512 Block Diagram	37
Figure 2-2 QLM Grouping	40
Figure 2-3 TWSI Addressing Format	41
Figure 2-4 GPIO Interface Assignment	43
Figure 2-5 WANic-66512 Clocking	47
Figure 2-6 Synchronous Ethernet Clock	48
Figure 2-7 FPGA Function Block Diagram	49
Figure 2-8 Mini-RDIMMs on WANic-66512	53
Figure 2-9 USB Flash Drive Header	55
Figure 2-10 10G PHY Implementation	56
Figure 2-11 Header Locations	60
Figure 2-12 JTAG Scan Chain	63
Figure 2-13 S1 DIP Switch Approximate Location	64
Figure 3-1 GPIO14 Interrupt Scheme	69
Figure 3-2 FPGA Revision Register @ Base + 00h	70
Figure 3-3 Board ID and DIP Switch Register @ Base + 01h	71
Figure 3-4 LED Control Register @ Base + 02h	72
Figure 3-5 Interrupt Control Register @ Base + 03h	73
Figure 3-6 Interrupt Status Register @ Base + 04h	74
Figure 3-7 Peripheral Reset Register @ Base + 05h	75
Figure 3-8 Transmit Control Register @ Base + 06h	76
Figure 3-9 Fan and Temperature Status Register @ Base + 07h	77
Figure 3-10 I2C Control Register @ Base + Device Address	80
Figure 3-11 I2C Address Low Register @ Base + Low Address	81
Figure 3-12 I2C Address High Register @ Base + High Address	81
Figure 3-13 I2C Data Register @ Base + Device Address	82
Figure 3-14 Boot Flash Upper Address Control Register @ Base + 20h	83
Figure 3-15 Miscellaneous Functions Register @ Base + 21h	84
Figure 3-16 PHY Interrupt Register @ Base + 22h	85
Figure 3-17 Thermal Interrupt Register @ Base + 23h	86
Figure 3-18 IR3541 Interrupt Register @ Base + 24h	87
Figure 3-19 DIMM Thermal Register @ Base + 25h	88
Figure 4-1 Software Components	89
Figure 4-2 S1 DIP Switch Location	97
Figure 4-3 EEPROM Mapping EEPROM Tuples	107
Figure 4-4 Flash Mapping	130

Figure 4-5 NPA Driver	134
Figure 4-6 IOCTL Commands and Structures	136
Figure 4-7 IOCTL Command Types	140
Figure 4-8 Exported Kernel Symbols	143
Figure 4-9 Example of the NPLSP Configuration Menu	145
Figure 5-1 Command Line Parameters	152
Figure 5-2 OCTEON NPA Major Number	153
Figure 5-3 Diagnostic Utility Main Menu	154
Figure 5-4 View Test Results Display	156
Figure 5-5 EEPROM Menu	157
Figure 5-6 Run Auto Test	158
Figure 5-7 Test Configuration with Cable Attachment	159
Figure 5-8 Auto Test Configuration Menu	159
Figure 5-9 Diagnostic LED Location	161
Figure 5-10 Flash Menu	162
Figure 5-11 Log File Location	163
Figure 5-12 Show Core Temperature	164
Figure 5-13 Memory Test Results	165
Figure 5-14 Memory Test with Command Line Parameter -a	165
Figure 5-15 USB Storage Test	166
Figure 5-16 Error Message Display	167

List of Tables

Table 1-1 WANic-66512 Models	15
Table 1-2 Environment Requirements	19
Table 1-3 WANic-66512 Minimum Volumetric Air Flow	20
Table 1-4 WANic-66512 Current and Power Limits	22
Table 1-5 Supported Optical SFP+ Transceivers	25
Table 1-6 Link Status LEDs	26
Table 1-7 Power Status LEDs	26
Table 2-1 Chip Select Space	39
Table 2-2 Local TWSI/I2C Device Addresses	42
Table 2-3 GPIO Interface Bits	44
Table 2-4 CIU_WDOG[0:5] Register Description	46
Table 2-5 CIU_PP_POKE{0:5} Register Description	46
Table 2-6 FPGA TWSI Devices and Addresses	51
Table 2-7 USB Flash Header Pinout	55
Table 2-8 PCIe Slot Power Limits	57
Table 2-9 JTAG Mode S1 DIP Switch Configuration	65
Table 2-10 Boot Mode S1 DIP Switch Configuration	65
Table 2-11 Diagnostic Mode S1 DIP Switch Configuration	65
Table 3-1 FPGA Memory Mapped Registers	67
Table 3-2 Build Configuration Bit Values	71
Table 3-3 Diagnostic LEDs on WANic-66512	72
Table 4-1 Linux Applications and Utilities	92
Table 4-2 POST Failures	96
Table 4-3 U-Boot Commands	98
Table 4-4 Environment Variables	103
Table 4-5 Load and Boot Image Default Entries	106
Table 4-6 EEPROM Tuples	108
Table 4-7 Flash Driver Functions	129
Table 4-8 Flash Partitioning	131
Table 4-9 IOCTL Command Structures	137
Table 5-1 Command Line Parameters	152
Table 5-2 Diagnostic Utility Main Menu	155
Table 5-3 EEPROM Menu Options	157
Table 5-4 Test Configuration Menu Options	160
Table 5-5 Diagnostic LED Test Prompts	161
Table 5-6 Flash Menu Options	162
Table 5-7 In-Service Daemon Options	167
Table 6-1 Hardware Specifications	169
Table 6-2 Regulatory Compliance	170

Preface

This document provides technical information for the WANic-66512 Packet Processor. The WANic-66512 is a PCI Express (PCIe) Generation 2 standard-height, half-length adapter card that offers two 10 Gigabit Ethernet (GbE) front panel I/O interfaces.

Purpose

The purpose of this manual is to describe the WANic-66512 and the services the card provides. This manual includes a description of the WANic-66512 hardware and firmware, as well as information for using the accompanying Linux device drivers.

Audience

This manual is intended for the user who creates and develops applications for the WANic-66512 such as an engineer or developer.

It is assumed that the user is familiar with:

- Standard cabling
- Configurations of operating systems, networks, and PCI Express technology
- C programming with the Linux Operating System

Referenced Documentation

Documentation referenced in this manual includes the following industry and vendor documentation.

Industry Standard Specifications

- *IEEE® 802.3–Standard for Information Technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements–Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, Institute of Electrical and Electronics Engineers (IEEE).
- *PCI Express® Base Specification R1.1*, PCI Special Interest Group (PCI-SIG®).
- *PCI Express® Card ElectroMechanical Specification R1.1*, PCI-SIG.
- *SFF-8431 Specifications for Enhanced Small Form Factor Pluggable Module (SFP+)*, Multi-Source Agreement.

Vendor Documentation

- *IR3541 Digital Multi-Phase Buck Controller* datasheet, International Rectifier,® Inc.
- *IR3550 60A Integrated PoweIRstage®* datasheet, International Rectifier, Inc.
- *OCTEON® Plus CN54/5/6/7xx Hardware Reference Manual*, Cavium® Inc.
- *QT2025™ PxKD Serial 10Gbps-to-XAUI Transceiver with Adapter EDC Data Sheet*, Applied Micro Circuits Corporation®.
- *ZL30152, Universal Rate Adapting Synchronous Clock Generator* datasheet, Zarlink™ Semiconductor, Inc.

Manual Organization

This manual contains the following chapters and appendices:

- ***“Chapter 1: Getting Started,”*** provides precautions and instructions for installing the WANic-66512 Packet Processor.
- ***“Chapter 2: Hardware Description,”*** describes the hardware components on the WANic-66512 Packet Processor. This chapter also includes a description of connectors, pinouts, and Light Emitting Diodes (LEDs).
- ***“Chapter 3: FPGA Registers,”*** describes the registers to use for communications between components on the WANic-66512 Packet Processor.
- ***“Chapter 4: Software Description,”*** provides information on how to use software components including a bootloader, Application Programming Interface (API), and drivers.
- ***“Chapter 5: Linux Support Package (LSP) Applications,”*** describes the Linux Software Package (LSP), which includes the Diagnostic Utility and the In-Service Daemon. The Diagnostic Utility verifies and configures components on the WANic-66512 Packet Processor. The In-Service Daemon monitors hardware continuously and reports detected faults.
- ***“Chapter 6: Specification,”*** provides a listing of hardware specifications and regulatory compliance information.
- ***“Appendix A • GE Intelligent Platforms, Inc. Software License Description,”*** identifies the applicable software licenses for select WANic-66512 Packet Processor software.
- ***“Appendix B • GNU General Public License V2,”*** contains a copy of the GNU General Public License.

This manual also contains a glossary and an index.

Notation Conventions

This manual uses the following notation conventions:

- *Italics* emphasizes words in text, register field names, and documentation or chapter titles.
- Hexadecimal values are represented as digits followed by “h”, for example 0Ch.
- Courier text identifies text that the user must enter or text that displays on the screen. For example, “Use the `ioctl DevShow` command to verify the driver installation.”
- **Bold Palatino Linotype text** identifies register names. For example, “An **Interrupt Control Register** is available for each I/O bus space.”
- **Bold Courier text** identifies text in an example that the user must enter. For example, “Load the driver manually by typing the `modprobe` command:
`modprobe -v xxxx`”
- Notes, cautions, and warnings call attention to essential information:



NOTE

A Note calls attention to important information, such as tips and advice.



CAUTION

A Caution alerts you to conditions that could damage a device, system, or data.



WARNING

A Warning calls attention to actions that can cause risk of personal injury.

- Specific term definitions, as applied in this manual include:
 - “May” means that there is flexibility that does not affect the outcome or result of the action.
 - “Should” means that the user has flexibility but it is strongly recommended to perform the specified action to achieve an optimal outcome or result.
 - “Must” means that there is no flexibility and the user is required to perform the action to achieve an optimal outcome or result.

Warranty and Repair

GE Intelligent Platforms, Inc. provides a comprehensive web site on the World Wide Web (<http://www.ge-ip.com>.) This web site contains up-to-date information including current and new products, such as the WANic family of packet processors. The web site also contains sales office locations, copyrights, trademarks, press releases, warranties, and technical support information.

Warranty

Warranty information is described on the GE Intelligent Platforms Embedded Systems web site at <http://www.ge-ip.com/support/embeddedsupport/warranty>. This site provides current product warranty and repair services as well as information on out-of-warranty services. For additional information on specific product warranty, contact Customer Technical Support or your local sales representative.

Customer Technical Support

GE Intelligent Platforms' dedicated team of Customer Technical Support Engineers is committed to providing quality support to all their customers. Customer Technical Support Engineers are trained to assist GE Intelligent Platforms customers in the development, integration, and use of GE Intelligent Platforms products in customer applications, systems, and products to facilitate timely product development. The Customer Technical Support Service Center is staffed weekdays (except holidays) between the hours of 8:00 AM and 5:00 PM Eastern Standard Time (CST).

Use one of the following methods to contact technical support:

Email:	support.embeddedsystems.ip@ge.com
Telephone:	1-800-433-2682
Address:	GE Intelligent Platforms 12090 South Memorial Parkway Huntsville, AL 35803-3308
Hours:	8:00 AM to 5:00 PM (CST)

Before contacting technical support, make sure you have the following information available:

- Model
- Purchase receipt
- Description of problem

When a product must be returned, contact Customer Technical Support for a Return Material Authorization (RMA) number. The RMA number must be obtained *prior* to any return so that it can be referenced in all communications.

1 • Getting Started

The WANic-66512 Packet Processor (hereafter referred to as the WANic-66512) is a dual-port 10 Gigabit Ethernet (GbE) adapter card that features Cavium's[®] OCTEON[®] II CN6645 Multi-Core Processor. The WANic-66512 provides two Small Form-factor Pluggable Plus (SFP+) housing bays suitable for connecting to Direct Attach Cable (DAC) cable assembly, or fiber optic modules supporting SR/LR fiber optic cabling. The OCTEON II CN6645 is a 10 core processor that features an Application Accelerator Processor (AAP), which supports a multitude of capabilities including:

- Security
- RAID storage technology
- Encryption information transformation
- Regular Expression (RegEX) acceleration
- Networking
- Transmission Control Protocol (TCP) acceleration
- Compression/decompression
- De-duplication
- Quality of Service (QoS) performance

The WANic-66512 is a high-performance, dual-port 10 Gigabits per sec (Gb/s) intelligent PCI Express (PCIe) adapter card that complies with PCIe 2.0 (Generation 2) and 1.1 specifications.

All WANic-66512 cards have been tested for interoperability, industry certification and regulatory compliance to ensure compatibility and rapid application deployment. A version of the WANic-66512 is also designed for NEBS compliance. In addition, all WANic-66512 cards meet the European Union Restriction of Hazardous Substance (RoHS) directive.

The WANic-66512 provides the two models listed in Table 1-1, which are configured at the factory. See [Chapter 2: Hardware Description](#) for a description of featured components.

Table 1-1 WANic-66512 Models

WANic-66512 Models	cnMIPS Cores	Processor Speed (GigaHertz)	DDR3 Memory (GigaBytes)	eUSB Memory (GigaBytes)	Operating Temperature ¹
WPC6D74010A	10	1.3GHz	4GB	2GB	0°C to +55°C
WPC6D84010B	10	1.5GHz	4GB	2GB	0°C to +50°C

1. See [Section 1.2 "Environmental Requirements"](#).



NOTE

Unless specifically specified otherwise, the information in this manual applies to both WANic-66512 models.

1.1 Handling Precautions

Before handling any GE Intelligent Platforms component, read the following Electrostatic Discharge (ESD), laser/optical, and safety precautions.



WARNING

GE Intelligent Platforms, Inc. assumes no liability for the user's failure to comply with required ESD and safety precautions.

1.1.1 Electrostatic Discharge (ESD) Precautions

Always take necessary precautions to prevent ESD. ESD can damage the WANic-66512, which has ESD sensitive components.

All products must be in an anti-static plastic bag or conductive foam for storage or shipment. The user or installer must work at an approved anti-static work station when unpacking the WANic-66512. The user or installer must *always* use anti-static grounding straps to prevent damage due to ESD.

Always use the following precautions to prevent ESD when installing or removing the WANic-66512.



CAUTION

Always ground yourself *before* touching the board or module. Ground yourself by either touching a grounded unpainted metal surface or by using an ESD-protective wrist strap from your wrist to a bare, unpainted metal section of the chassis. This prevents ESD from damaging the product.

When working with a chassis that cannot be placed upon a grounded antistatic mat, connect a grounding strap between the electrical input ground and the facility electrical service ground.

Always keep the WANic-66512 in its static-protective packaging when it is not installed in the system.

Always hold the WANic-66512 by its handles or edges. Avoid touching the components or connections on the board.



WARNING

Depending on the chassis, open equipment enclosures and chassis can expose hazardous voltage, which may cause electric shock to the installer or user. Make sure line power to the equipment is disconnected before servicing the chassis and components. Refer to the manufacturer's documentation that accompanies the chassis.

1.1.2 Optical/Laser Precautions

For optical/laser devices, read the following notes, cautions, and warnings:



WARNING

1. For this product, use only the Class 1 laser devices that have the following approvals:
 - FDA 21 CFR 1040.10
 - IEC 60825-1
2. An optical SFP+ transmitter is a Class 1 device. Invisible laser radiation may be emitted from disconnected fibers or connectors. Do not stare into beams or view directly with optical instruments as this may permanently damage your eyes.



CAUTION

1. It is important to disconnect or remove all cables before removing or installing an optical SFP+ transceiver. Failure to do so may result in damage to the cable or SFP+ device.
2. Do not leave an optical SFP+ transceiver uncovered except when inserting or removing a cable. The safety/dust plugs keep the port clean and prevent accidental exposure to laser light.



NOTE

Protect optical SFP+ modules by inserting clean dust plugs into the SFP+ modules after the cables are extracted from them. Be sure to clean the optic surfaces of the fiber cables before plugging the dust plugs back into the optical bores of another SFP+ module. Avoid getting dust and other contaminants into the optical bores of your SFP+ modules: The optics will not work correctly when obstructed with dust.

1.1.3 Safety Precautions

Observed the following safety precautions during all phases of the operation, service, and repair of this product. Failure to comply with these precautions, or with specific warnings elsewhere in this manual, violates safety standards of design, manufacture and intended use of this product.

- **Ground the System**

To minimize shock hazard, the chassis and system cabinet must be connected to an electrical ground.

Refer to the documentation that accompanies the chassis or system cabinet for specific grounding information.

- **Do Not Operate in an Explosive Atmosphere**



CAUTION

Do not operate the system in the presence of flammable gases or fumes. Operation of any electrical system in such an environment constitutes a safety hazard.

- **Keep Away from Live Circuits**

Operating personnel must not remove product covers. Component replacement and internal adjustments must be made by qualified maintenance personnel. Do not replace components with power cable connected. Under certain conditions, dangerous voltages may exist even with the power cable removed. To avoid injuries, always disconnect power and discharge circuits before touching them.

- **Do Not Substitute Parts or Modify System**



CAUTION

Do not install substitute parts or perform any unauthorized modification to the product. Return the product to GE Intelligent Platforms for service and repair to ensure that safety features are maintained.

1.2 Environmental Requirements

It is important to follow the environmental requirements identified in this section. The WANic-66512 was designed and tested to perform with the specifications in Table 1-2.



WARNING

Do not operate the WANic-66512 outside the specifications listed in Table 1-2. Failure to comply with these specifications may result in damage to the board. Therefore, GE Intelligent Platforms, Inc. shall not be held liable for any damages caused directly or indirectly by use of the WANic-66512 outside the scope and specifications defined in this manual.

Table 1-2 Environment Requirements

Environmental Requirements	Model - WPC6D7401A (1.3 GHz Processor)	Model - WPC6D8401B (1.5 GHz Processor)
Operating Temperature ¹	−5°C to +55°C (+23°F to +131°F)	−5°C to +50°C (+23°F to +122°F)
Storage Temperature	−40°C to +85°C (−40°F to +185°F)	−40°C to +85°C (−40°F to +185°F)
Relative Humidity	5% to 90% non-condensing	5% to 90% non-condensing
Operating Altitude	4,572 meters (15,000 feet)	4,572 meters (15,000 feet)
Operating Random Vibration	5 – 500Hz, 1G axis	5 – 500Hz, 1G axis
Operating Mechanical Shock	5G each axis	5G each axis

1. Operating Temperature describes the air temperature that circulates around the WANic-66512 within a PCI Express chassis.

See [Chapter 6: Specification](#) for hardware specification and regulatory compliance information.

1.3 Air Flow and Cooling Requirements

The WANic-66512 is designed for an air-cooled chassis environment. The WANic-66512 thermal requirements are satisfied by a heat sinks and a fan.

Use the Temperature Sensor to monitor the Processor and to determine whether there is sufficient cooling to maintain a Processor junction temperature of less than 110°C in accordance with the maximum ratings specified for the Processor. See [Section 2.9 "Temperature Management"](#) in [Chapter 2: Hardware Description](#) for more information on temperature monitoring.



WARNING

1. The WANic-66512 shall not be used in an environment with a temperature higher than that specified in Table 1-3. While hardware protection against thermal runaway exist in the WANic-66512, GE Intelligent Platforms shall not be held liable for any damages caused directly or indirectly by use of the WANic-66512 outside the scope and specifications defined in this manual.
2. Some systems reduce airflow automatically based on the host CPU temperature sensors, and do not provide the WANic-66512 with the required airflow, which may result in permanent damage to the WANic-66512.

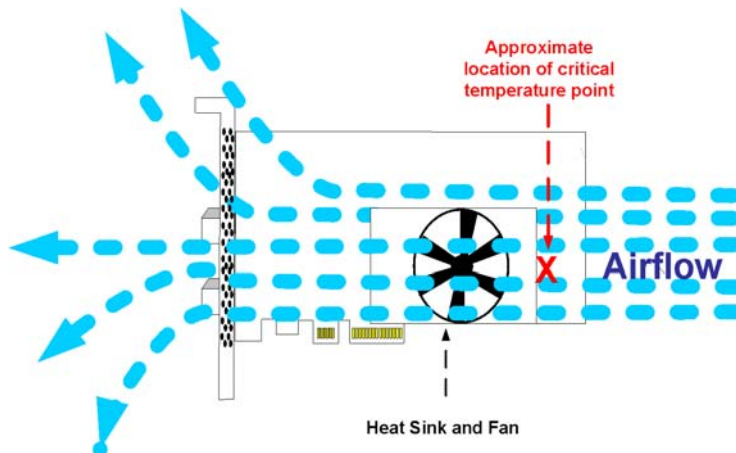
The WANic-66512 must be installed into a system that consistently provides the minimum volumetric air flow in Linear Feet per Minute (LFM) as listed in Table 1-3.

Table 1-3 WANic-66512 Minimum Volumetric Air Flow

Model	WPC6D74010A (1.3 GHz Processor)	WPC6D84010B (1.5 GHz Processor)
Board with Fan Assembly	50 LFM @ 25° C 275 LFM @ 40° C 275 LFM @ 55° C	50 LFM @ 25° C 300 LFM @ 40° C 300 LFM @ 55° C

The air flow path is similar to that shown in Figure 1-1 depending on the location of air vents within the chassis.

Figure 1-1 WANic-66512



1.4 Power

The WANic-66512 hardware uses multiple voltage sources. The WANic-66512 hardware is powered through two main supplies:

- PCI Edge Fingers (+3.3V)
- J4 PCIe External Graphics Power Connector (+12V) located on the upper corner of the board as shown in Figure 1-2. The pinout for this connector is provided in Figure 1-2.



NOTE

For more information on power distribution, see [Section 2.8 "Power Distribution"](#) in Chapter 2.

Instructions for connecting to the J4 PCIe Graphic Power Connector are provided in this chapter in [Section 1.9 "Hardware Installation"](#).

Figure 1-2 WANic-66512 External Power Connector

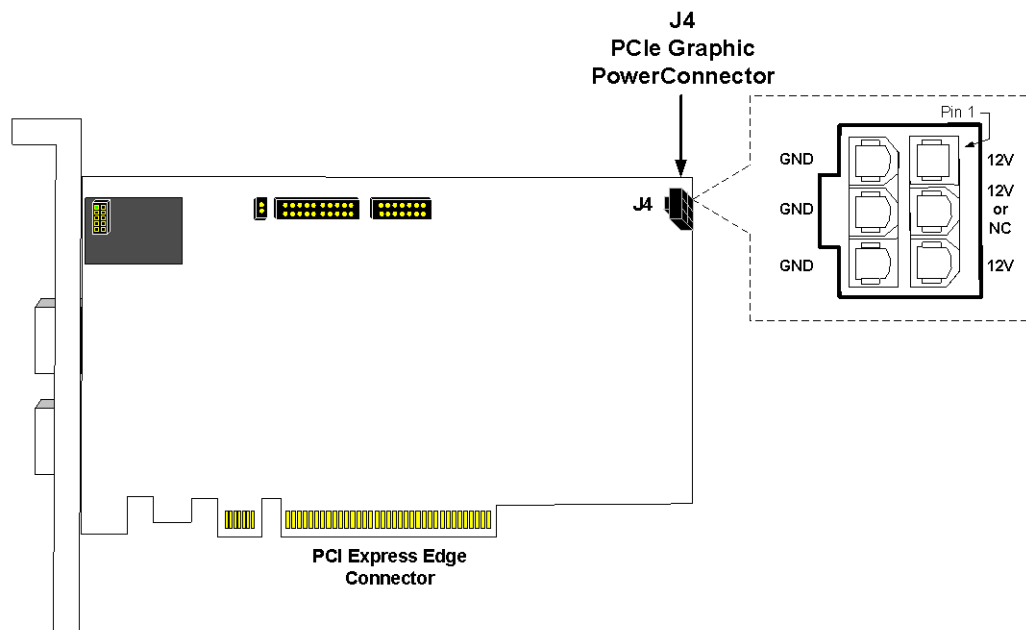


Table 1-4 identifies the current and power supply limits for both connectors. Typical user power will vary.

Table 1-4 WANic-66512 Current and Power Limits

Rail Voltage (V)	Source	WPC6D7401A (1.3 GHz Processor)	WPC6D8401B (1.5 GHz Processor)
		Typical Maximum Watts (W)	
+3.3V	PCIe Edge	5.2 W	5.2 W
+12V	J4 External Power	48.0 W	56.1W
		Total: 53.2 W	Total: 61.3 W

**WARNING**

It is important to maintain proper air flow and cooling. See [Section 1.3 "Air Flow and Cooling Requirements"](#) for more information.

1.5 Required Tools

A small slotted screwdriver is required to secure the front panel retaining screws.

An ESD wrist or boot strap, or an ESD anti-static mat or device is necessary when handling the WANic-66512.

1.6 Installation Requirements

The WANic-66512 requires the following components for installation, which are purchased separately:

- PCI Express-compliant chassis
- SFP+ modules
- J4 PCI External Graphic Power Connector cable and mating connector assembly

1.7 Unpacking Procedures

Always observe any precautions found in the shipping container.

- Carefully unpack and thoroughly inspect all items. Refer to the packing slip to verify that all items are present.
- Inspect each item for damage that might have occurred during shipment. The item(s) should be checked for broken components, damaged printed circuit board(s), heat damage or other visible contamination. All claims arising from shipping damage should be filed with the carrier and a complete report sent to GE Intelligent Platforms Customer Technical Support.
- Save the packaging for future use if necessary.

The WANic-66512 is configured at the factory and shipped with factory settings. Options can be configured using the installation procedure. Configuration changes can be set with control registers (see [Chapter 3: FPGA Registers](#)) after the WANic-66512 is installed in a system. Use the WANic-66512 Software CD-ROM to install the WANic-66512 software.

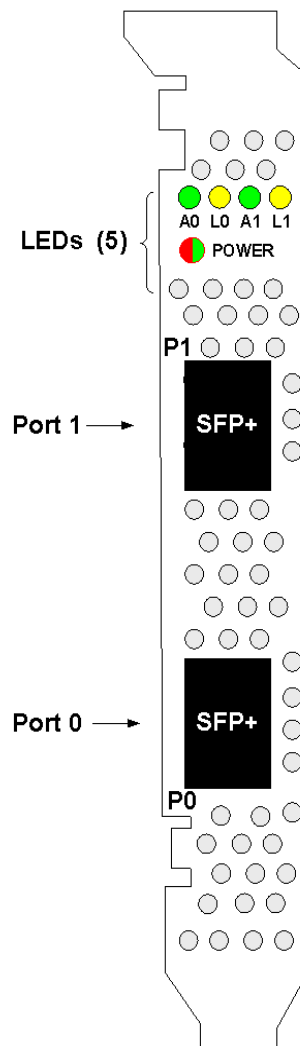
1.8 WANic-66512 Front Panel

Before attempting the installation of the WANic-66512 adapter board, become familiar with the WANic-66512 front panel. The WANic-66512 front panel connectors and LEDs are shown in Figure 1-3. (There are also numerous venting holes throughout the front panel.)

As shown in Figure 1-2, the WANic-66512 front panel provides the following:

- Two SFP+ ports
- Five LEDs, including:
 - One Power LED (red/green)
 - Four port Activity and Link Status LEDs, including:
 - Two green Activity (A[1:0]) LEDs
 - Two yellow Link (L[1:0]) LEDs

Figure 1-3 WANic-66512 Front Panel



1.8.1 SFP+ Housing Bays

SFP+ housing bays provide flexibility to the type of media you can use for I/O. The WANic-66512 supports fiber optic SFP+ transceivers and Direct Attach Mode cable assembly, which are *optional* components purchased separately.

The WANic-66512 front panel contains two SFP+ ports labeled P0 through P1. Each port has a corresponding green LED described in [Section 1.8.2 "LEDs"](#).



NOTE

Contact a GE Intelligent Platforms' Customer Technical Support Engineer for more information on supported SFP+ transceivers.

1.8.1.1 Optical SFP+

SFP+ optical/fiber transceivers support 10GBASE-SX/LX connectors. Supported SFP+ models are listed in Table 1-5, which are optionally available from GE Intelligent Platforms (or another vendor).

Table 1-5 Supported Optical SFP+ Transceivers

Model	Description
SFP-0A	10 Gigabit Ethernet SFP+ optical transceiver module for Multi-mode fiber 850-nm wavelength.
SFP-0B	10 Gigabit Ethernet SFP+ optical transceiver module for Single-mode fiber, up to 10Km.

1.8.1.2 Direct Attach Mode

The Direct Attach Mode cable assembly is an optional component and is used in conjunction with the WANic-66512 software to provide Direct Attach Mode on the WANic-66512. Direct Attach Mode, as defined by the Multi-Source Agreement Group in the SFF-8431 specification, detects when a cable is attached to the SFP+ connector and sets up the interface for the SFP+ automatically. The Direct Attach Mode cable assembly can be purchased separately from GE Intelligent Platforms (Part No. 42G7690-0003).

Refer to <http://www.sffcommittee.org/ie/sffspec.html> for more information on the SFF-8431 specification.

1.8.2 LEDs

The front panel contains five LEDs as shown in Figure 1-3.

Link and Activity Status LEDs

Table 1-6 describes the two Link and two Activity status LEDs for Port 0 and Port 1.

Table 1-6 Link Status LEDs

LED	Color	Status Indicator	State	Description
A[1:0]	Green	Activity	Off	No activity.
			On/Blinking	Link is transmitting and receiving.
L[1:0]	Yellow	Link	Off	Link is down and not working properly.
			On/Blinking	Link is up and working properly

Power LED

The Power LED provides voltage/power supply indications for the WANic-66512.

Table 1-7 Power Status LEDs

LED	Color	Status Indicator	State	Description
POWER	Green	Power	On	Normal operation, power supplies okay.
	Red		On	Thermal event occurred. Hard reset is required. ¹

1. Power fault occurs when one of the self monitoring voltages experiences a failure causing an interruption in the sequential process. See **Section 2.12 "Power"** for additional information.

1.9 Hardware Installation

This section describes how to install or remove the WANic-66512 within a PCIe chassis. It also includes directions for insertion and removal of an SFP+ module.

Before working with any GE Intelligent Platforms component, take the necessary precautions to prevent ESD. (See [Section 1.1 "Handling Precautions"](#).)

1.9.1 Installation Overview

Installing a WANic-66512 involves the following operations:

1. Install SFP+ modules.
2. Install the WANic-66512:
3. Install software.

1.10 SFP+ Modules Installation and Removal

SFP+ modules can be inserted and removed in the front panel of a WANic-66512 card before or after the card is inserted in a chassis.

Hot swapping an SFP+ module on a WANic-66512 is supported only when using the GE Intelligent Platforms BSP (npaDriver) running in Linux.



CAUTION

U-Boot **does not** support hot swapping of an SFP+. If an SFP+ is hot swapped while in U-Boot, you must run the `sfpmnit` U-Boot command from the U-Boot prompt to re-initialize all SFP+ ports after U-Boot is installed. See '[Section 4.4.3 U-Boot Commands](#)' for a listing of U-Boot commands.

For fiber optical modules, please observe the following warnings, cautions, and notes:



WARNING

For this product, use only the Class 1 laser device that have the following approval:

- FDA21 CFR 1040.10
- IEC 60825-1

Invisible laser radiation may be emitted from disconnected fibres or connectors. Do not Stare into beams or view directly with optical instruments as this may permanently damage your eyes.



CAUTION

1. It is important to disconnect or remove all cables before removing or installing an optical SFP+ transceiver. Failure to do so may result in damage to the cable or SFP+ device.
2. Do not leave an optical SFP+ transceiver uncovered except when inserting or removing a cable. The safety/dust plugs keep the port clean and prevent accidental exposure to laser light.



NOTE

Protect optical SFP+ modules by inserting clean dust plugs into the SFP+ modules after the cables are extracted from them. Be sure to clean the optic surfaces of the fiber cables before plugging the dust plugs back into the optical bores of another SFP+ module. Avoid getting dust and other contaminants into the optical bores of your SFP+ modules: The optics will not work correctly when obstructed with dust.

1.10.1 SFP+ Modules Installation

To insert an SFP+ module into the SFP+ housing bay on the WANic-66512 front panel, perform the following steps:

1. Ground yourself or attach an ESD-preventive wrist strap from your wrist to a bare metal surface of the chassis.
2. Remove power from the adapter.
3. Remove the SFP+ module from the anti-static packaging.



NOTE

SFP+ modules have different socket configuration. Make sure you have the proper orientation for the SFP+ you are inserting.

4. Remove dust protectors and save them for future use.
5. Align the SFP+ module to the front of the SFP+ housing bay.



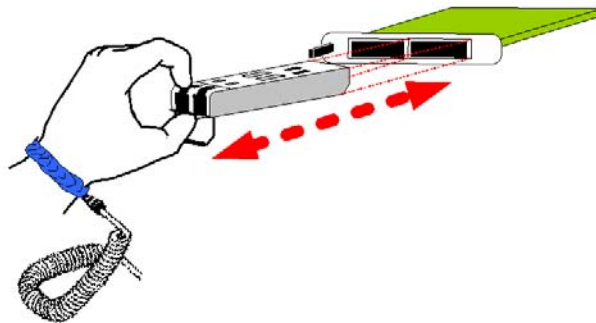
NOTE

Since SFP+ modules have various latch designs, follow the instructions provided with the SFP+ for inserting the transceiver properly.

Glide the SFP+ module into the bay as shown in Figure 1-4. Be sure to feel or hear the module snap into place.

If the SFP+ module contains a clasp, close the clasp by moving the clasp up and then pressing it firmly into the locked position.

Figure 1-4 Inserting or Removing an SFP+ Module.



6. Insert the cable connector into the SFP+ module
7. Return power to the board.

1.10.2 SFP+ Module Removal

To remove an SFP+ module from a WANic-66512, perform the following steps:

1. Ground your self or attach an ESD-preventive wrist strap from your wrist to a bare metal surface of the chassis.
2. Remove power from the board.
3. Disconnect the cable from the SFP+ module.

For optical SFP+ transceivers, insert the dust plugs into the SFP+ transceiver immediately.

4. Unlock and remove the SFP+ module.

Since SFP+ modules have various latch designs, follow the instructions provided with the SFP+ for removing the transceiver. For example, on some modules, you can open the latch by pressing it in the appropriate direction with your index finger or a long narrow tool such as a flat-blade screw driver.

5. Grasp the SFP+ module between your thumb and index finger. Then, carefully slide the SFP+ out of the SFP+ housing bay on the adapter.
6. Insert dust protectors into the SFP+ transceivers.
7. Store the SFP+ transceiver in static-protective packaging.
8. Return power to the board.

1.11 WANic-66512 Installation and Removal

The WANic-66512 inserts into a PCIe slot with a minimum of four lanes.

Also, the WANic-66512 contains a J4 Connector, which must be attached to an external power supply.

1.11.1 WANic-66512 Installation

The WANic-66512 is designed for an air-cooled chassis environment. Evaluate your system to make sure the WANic-66512 adapter is installed in a chassis that provides consistent and cooling as listed in [Chapter 1.3: Air Flow and Cooling Requirements](#).

To install the WANic-66512 into a PCIe chassis, perform the following steps:

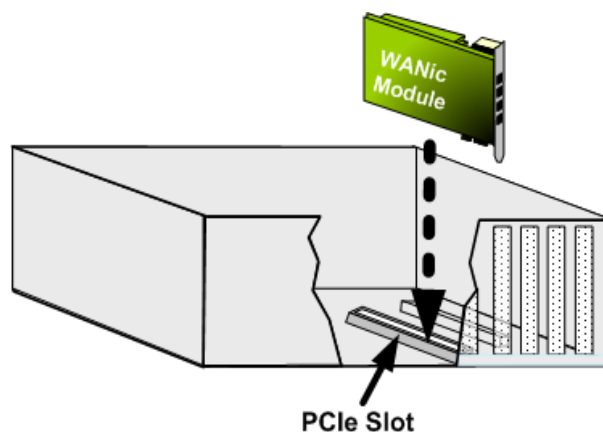
1. Ground yourself before handling the card. (*See* Section 1.1 "Handling Precautions")
2. Turn *off* power to the computer.
3. Disconnect the cables from the back of the computer chassis.
4. Remove the cover from the computer chassis.
5. Locate the available slot in the chassis.
6. If appropriate, remove the screw that secures the filler panel to the card cage frame for these slots. Then, remove the filler panel.
7. Remove the adapter from its static-protective packaging. Do not touch the components on the module.

CAUTION

Ground yourself *before* handling the module. While the card is still in the static-protective envelope, hold the card by the edges with one hand while you slide the static-protective envelop off the card with the other hand. Save the package in case you need it for future use.

8. Align the adapter with the appropriate slot (see Figure 1-5) and insert the adapter into the slot. Slide the adapter into the slot gently, but firmly.

Figure 1-5 Installation in a PCI Bus Chassis



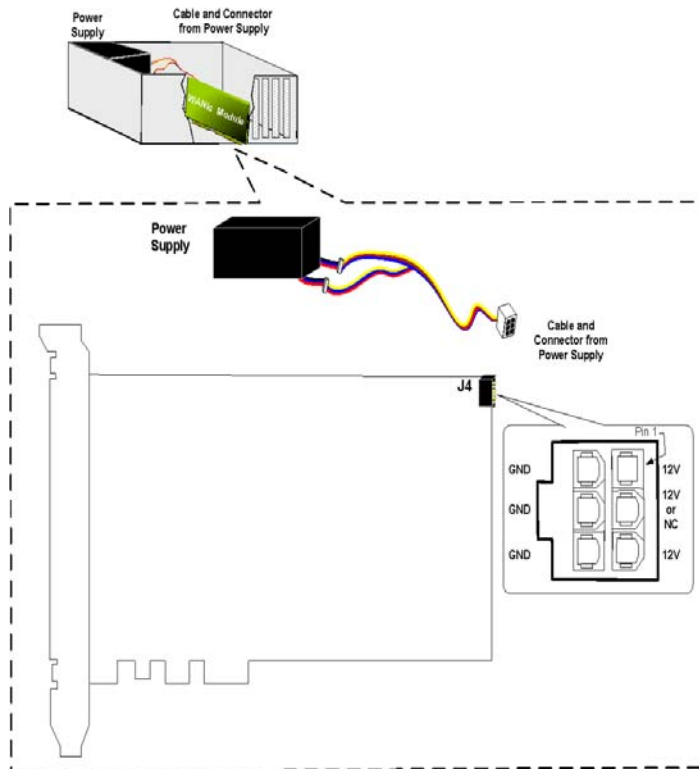
9. Attach the screw that secures the adapter to the card cage frame.
10. Attach the SFP+ to the adapter as described in *“Section 1.10 “SFP+ Modules Installation and Removal”*.
11. Attach the power cable assembly from an external power supply to the J4 Power Connector on the WANic-66512 as shown in Figure 1-6. The cable assembly is a 10 inch cable that splits one (LP6) power connector into two LP4 power connectors.



NOTE

Be sure to attach all three connectors.

Figure 1-6 Connecting to the Power Supply



12. Replace the computer chassis cover.
13. Replace the cables.
14. Power on the computer.
15. Run the diagnostic utility as described in *“Chapter 5: Linux Support Package (LSP) Applications.”*

1.11.2 WANic-66512 Removal

To remove the WANic-66512 from a PCIe chassis, perform the following steps:

1. Ground yourself before handling the WANic-66512.



CAUTION

Always ground yourself before touching the module. You can ground yourself by touching a grounded unpainted metal surface, such as a computer chassis. This prevents electrostatic discharge from damaging the module.

2. Turn *off* the power to the computer.
3. Disconnect the cables from the computer chassis.
4. Remove the cover from the computer chassis.
5. Identify the slot from which you intend to remove the adapter. Remove the screw that secures the adapter to the card cage frame.
6. Disconnect the connector from the power supply that is attached to the card.
7. Grasp the sides of the card. (Remember not to touch the components on the module.) Slide the card out of the slot.



NOTE

Due to the heat generated when the system is running, the adapter may feel warm when removing it. The adapter will cool if you wait several minutes before removing it.

8. Store the adapter in static-protective packaging. If you are replacing the adapter with a new WANic-66512, refer to [Section 1.11.1 "WANic-66512 Installation"](#). If you are not installing a new adapter, continue with Step 9.
9. Install the filler cards into the empty slots. Attach the screws that secures the filler cards to the card cage frame.
10. Replace the computer chassis cover.
11. Attach the cable, as appropriate.

1.12 Software Installation

The WANic-66512 Software Developer's Kit (SDK) provides an easy-to-use development interface for the Linux Operating System. The WANic-66512 provides boot-up services, diagnostics, and device drivers, as well as a customized Application Programming Interface (API) for select on-board devices.

WANic-66512 software on the CD-ROM consists of the following components as described in "*Chapter 4: Software Description.*"

- U-Boot Bootloader – loads and boots the WANic-66512 firmware.
- Linux Operating System – contains the Debian™ distribution with OCTEON Multi-Core Processor support.
- Linux Support Package (LSP) – also referred to as the NPLSP Package, provides a suite of functions to access and to use the WANic-66512 hardware including:
 - Diagnostic Utility
 - Debug Utility
 - In-Service Daemon
 - NPLSP Kernel Module
- Wind River PNE-LE Board Support Package (BSP) – provides a suite of functions to access and to use the WANic-66512 hardware.
- The PCI Driver Package – contains the component to configure and to manage the WANic-66512 hardware from a PCIe host, and to operate the card in either Network Interface Card (NIC) mode or Ethernet PCI mode

Appendix B: GNU General Public License V2 contains a copy of the *GNU General Public License*, version 2, for using and applying U-Boot, LSP, and the Cavium Software Developer's Kit.

1.12.1 Additional Software

For your convenience, the WANic-66512 CD-ROM also contains the following additional software:

- GNU Tool Chain – includes tools for building executables for the cnMIPS cores.
- GNU Debugger – is a standard debugger for the GNU software that helps diagnose a program that is executing.

Appendix B: GNU General Public License V2 contains a copy of the *GNU General Public License*, version 2 for using and applying these tools.

1.12.2 Operating System Support

The WANic-66512 supports the following operating systems:

- Linux Kernel 2.6.x Operating System, which is included in the software distribution.
- OCTEON Simple Executive or Generic Operating System, which the user must obtain from Cavium Networks, Inc. (www.caviumnetworks.com)
- Wind River PNE-LE Linux, Version 4.3, which the user must obtain from Wind River Systems, Inc. (www.windriver.com)

1.12.3 Cavium Networks Users' Organization Web Site

The Cavium Networks Users' website www.cnusers.org, provides a platform for expanding and supporting the user community for the OCTEON Multi-Core Processor (cnMIPS) family. This website provide a means for sharing software, such as Linux-based software, on the OCTEON Multi-Core Processor. This site includes data plane functions, drivers, and Simple Executive applications.



NOTE

Building applications that may use OCTEON functions require the Cavium Simple Executive (CVMX) libraries, which are provided in the Cavium Networks OCTEON SDK.

Refer to www.cnusers.org, for more information on specific applications.

2 • Hardware Description

This chapter describes the featured hardware components on the WANic-66512. A simple block diagram illustrates these featured components, which includes the Multi-Core Processor, memory devices, external interfaces, and connectors.

2.1 Overview

The WANic-66512 is a fully-integrated packet processor that provides flexible processing capabilities for development of telecommunications and other data processing intensive applications.

The WANic-66512 features a Cavium[®] OCTEON[®] II CN6645 Multi-Core Processor with 10 cnMIPS cores at speeds of up to 1.5 GigaHertz (GHz). Powerful Double Date Rate Type 3 (DDR3) Synchronous Dynamic Random Access Memory (SDRAM) with Error Correction Code (ECC) provides up to 4 GB (2 GB each slot) of packet memory (8GB is optional.)

The OCTEON II Multi-Core Processor provides a hybrid Nondeterministic Finite Automata/Deterministic Finite Automata (NFA/DFA) design called the *Hyper Finite Automata* (HFA) content processing engine. The HFA function delivers advanced pattern matching for Deep Packet Inspection (DPI) technology and is independent of pattern rule-set size and traffic flows.

The WANic-66512 supplies a Synchronous Ethernet clock sourcing feature, which synchronizes the clock at the Physical Layer to ensure that the clock timing is always derived from the most reliable clock source.

The WANic-66512 contains two front panel SFP+ bays for housing a flexible configuration of fiber optic 10 Gigabit Ethernet communications.

The WANic-66512 software installs seamlessly under popular Linux[®] distribution and includes Linux-based ready-to-use applications. The Linux Support Package (LSP) provides a suite of functions and drivers to access and to use the WANic-66512 hardware, and to provide a platform for the Control Plane.

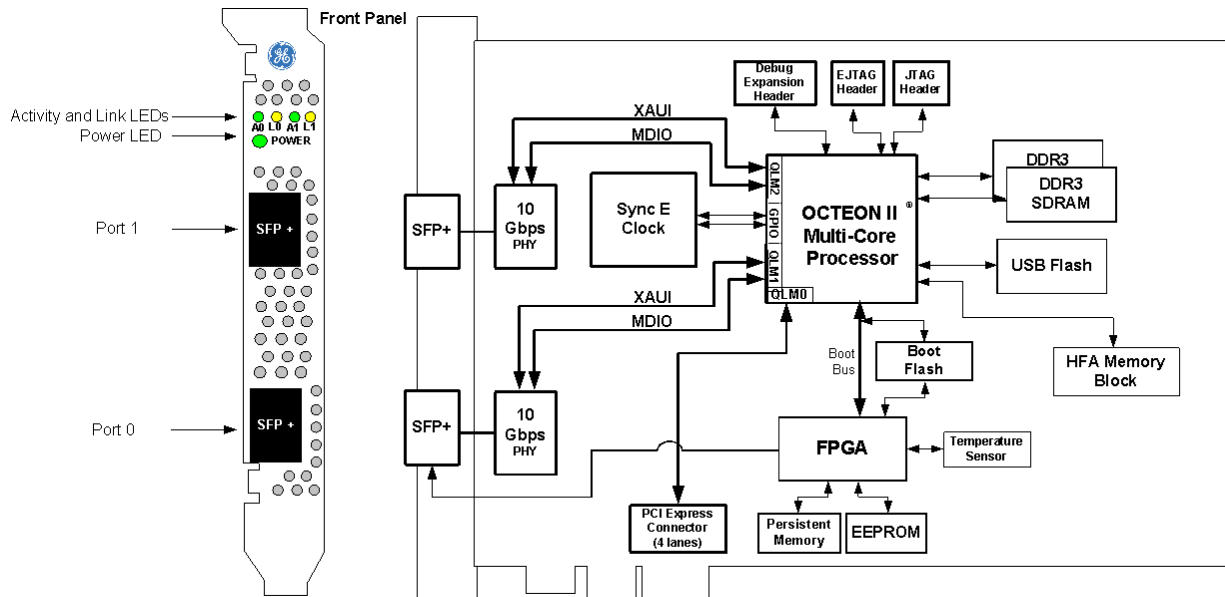
2.2 Features

The WANic-66512 features the following components:

- Cavium OCTEON Plus Multi-Core Packet Processor with 10 cores:
 - HFA/DFA deep packet inspection engine
 - Advanced Application Acceleration (AAA) Processor and Manager
 - 10 cnMIPS version II CPU cores with speeds up to 1.5 GHz
- Four (x4) lanes of PCI Express 2.0 to the host interface
- Two 10 GbE bays, which support the following in a non-mixed configuration:
 - Fiber/optical 10GBASE-LX/SX (single or multimode fiber) SFP+ transceivers
 - Copper 10GBASE-Cu transceivers
- Memory, including:
 - Up to 4 GB DDR3 in two VLP Mini-RDIMMs (2 GB per slot)
 - 512MB of DDR3 memory for HFA/DFA
 - 128 MegaBytes (MB) Flash for booting
 - 2GB eUSB removable flash disk for steady- state storage
 - 32MB Persistent Memory SDRAM
 - Serial EEPROM for general-purpose non-volatile data storage
- Field Programmable Gate Array (FPGA) for access to on-board components
- Synchronous Ethernet (SyncE) clock sourcing
- Front panel includes:
 - Light Emitting Diodes (LEDs) for module activity and link status indications
 - LED for Power Status indication
 - LED for Fan Fault indication
 - Two bays for two 10GbE SFP+ transceivers
- Debug headers via the optional Serial Debug Adapter:
 - OCTEON II Joint Test Action Group (JTAG) debug port
 - External JTAG (EJTAG) test/debug port
 - RS-232 console I/O port UART
- Temperature Sensor
- Linux 2.6.x Operating System support

The WANic-66512 features the components shown in a simple block diagram in Figure 2-1.

Figure 2-1 WANic-66512 Block Diagram



2.3 Multi-Core Processor

The WANic-66512 uses the Cavium OCTEON II (CN6645) family of packet processing Multi-Core Processors (hereafter referred to as the Processor). The Processor is a single-chip device for secure Open Systems Interconnection (OSI) Layer 2 to Layer 7 networking applications. The Processor provides many advanced features including:

- 10 cnMIPS cores
- HFA/DFA pattern matching
- Application Accelerator Processor (AAP)
- Networking
- Transmission Control Protocol (TCP) acceleration to optimize throughput
- Quality of Service (QoS) processing capability
- Encryption/decryption
- Compression/decompression

Each Processor is configured at factory assembly time to provide comprehensive integrated functions. The Processor is available with 10 cnMIPS cores with enhancements and additional built-in hardware acceleration for content and security processing. This architecture combines cnMIPS cores along with dedicated programmable coprocessor blocks capable of delivering up to 20 Gigabits per second (Gbps) of application performance at a core rate up to 1.5 GHz.

The Processor offers highly-flexible external networking interfaces. Interfaces include a four (x4) lane PCI Express (endpoint) interface and up to two 10 GbE ports.

- The PCI Express (PCIe) interface provides connectivity to the host.
- Two front panel 10 GbE bays allow housing for two 10 GbE transceivers for XAUI [Roman numeral X (10) – Attachment Unit Interface].

Packets and control information can flow to/from the Processor using any of the Processor's XAUI or PCIe interfaces.

The Processor uses external power to run a dedicated on-board power supply. This permits adjustments to the core device supply voltage, as needed, without affecting other circuitry.

The Processor uses the latest technology to implement the following:

- HFA/DFA Engine
- Boot Bus
- Quad Lane Module (QLM) packet interfaces
- Processor Interfaces, including:
 - 2 Two-Wire Serial Interfaces (TWSI)
 - 16 General Purpose Input/Output (GPIO) interfaces
 - 2 Universal Asynchronous Receiver Transmitter (UART) interfaces
 - 1 Board Power Supply interface
- Watchdog Timers
- Reference Clock

2.3.1 HFA/DFA Engine

The HFA/DFA engine is a pattern matching engine for DPI that has low latency, and is independent of the pattern rule-set size and traffic flows. The HFA/DFA engine features the following:

- High-performance regular expression pattern matching accelerator at speeds up to 1.5 GHz, independent of the number of rules or traffic flows.
- 4 Gbps+ DPI, with mainstream pattern memory and 10X+ reduction in graph size.
- Rule storage in DDR3 memory for HFA.
- Unique match length reporting.
- No limit on rule-sets and number of patterns in rule-set.
- Full compatibility with the Portable Operating System Interface for Unix (POSIX) and Perl Compatible Regular Express (PCRE) syntax.

2.3.2 Boot Bus

The WANic-66512 loads from the Processor Boot Bus (hereafter referred to as the Boot Bus). This big endian Boot Bus provides an 8-bit data path with:

- Eight Chip Selects
- Three multi-word DMA signals with dedicated DMA engines
- 32 address lines (using big endian byte-ordering)

When power-on reset completes, Core 0 of the Processor uses the Boot Bus to obtain code from the Flash memory device that is controlled by Chip Select 0 (CS0). The boot Flash memory device connects directly to the Boot Bus using 8-bit multiplexed mode and is controlled by Chip Select 0.

Chip Select 1 (CS1) provides access to FPGA registers.

Chip Select 2 (CS2) provides the lower 16 MB address of Persistent Memory (PMEM).

Chip Select 3 (CS3) provides the upper 16 MB address of PMEM.

Table 2-1 identifies each Chip Select.

Table 2-1 Chip Select Space

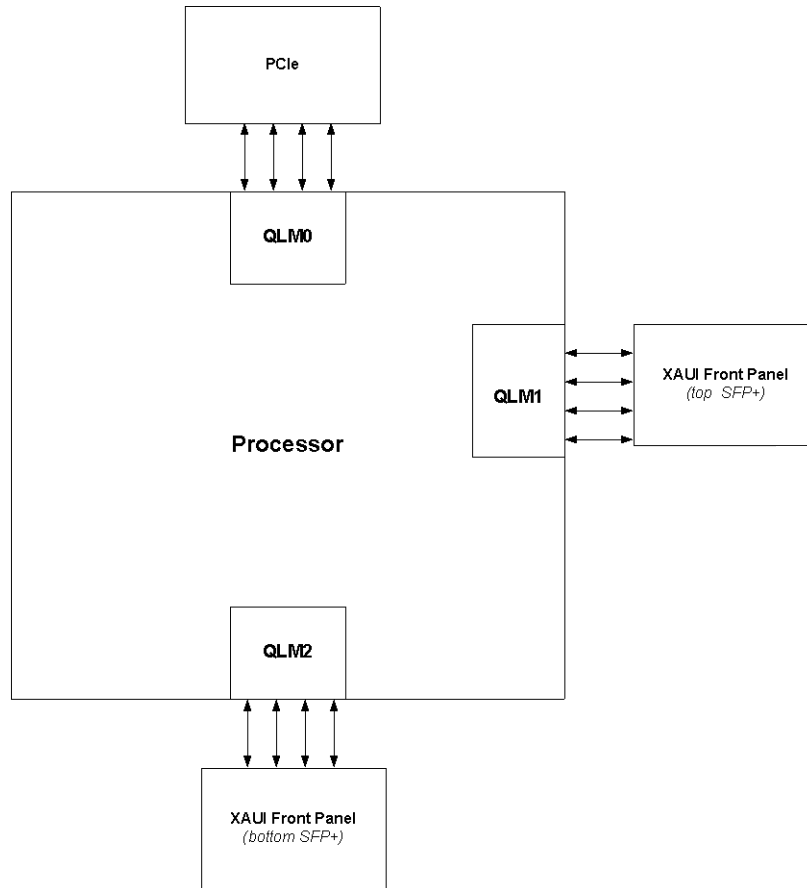
Chip Select	Address Range	Description
CS0	0x80010000_1FC00000 – 0x80010000_27BFFFFFF	Boot Flash
CS1	0x80010000_16000000 – 0x80010000_1600FFFF	FPGA
CS2	0x80010000_30000000 – 0x80010000_30FFFFFF	16 MB PMEM lower address space
CS3	0x80010000_31000000 – 0x80010000_31FFFFFF	16 MB PMEM upper address space

2.3.3 QLM Packet Interfaces

The Processor contains three Quad-Lane Modules (QLMs) for Serializer/Deserializer (SerDes) communications for a total of 12 SerDes lanes, grouped as shown in Figure 2-2. The WANic-66512 groups these QLMs on the Processor as follows:

- QLM0 – Four lanes of PCIe.
- QLM1 – XAUI1 (Port 1)) dedicated to the front panel (top) 10GbE SFP+ device.
- QLM2 – XAUI2 (Port 0) dedicated to the front panel (bottom) 10GbE SFP+ device.

Figure 2-2 QLM Grouping



- QLM0** QLM0 is a dedicated four lane PCIe endpoint interface (Ports 0–3). This four lane PCIe is a full-duplex interface that uses four self-clocked serial differential lanes in each direction to achieve up to 20 Gbps data throughput. Each lane operates at 2.5 or 5.0 Gbps to accommodate both data and overhead associated with 8B/10B coding. QLM0 provides all four PCIe lanes.
- QLM1** QLM1 provides a XAUI link dedicated to a front panel top 10GbE SFP+ transceiver and physical device (PHY).
- QLM2** QLM2 provides a XAUI link dedicated to the front panel bottom 10GbE SFP+ transceiver and PHY. (This is the SFP+ closest to the PCI Edge fingers.)

2.3.4 Processor Interfaces

The Processor contains the following additional interfaces:

- Two-Wire Serial Interface (TWSI)
- General Purpose I/O (GPIO)
- Universal Asynchronous Receiver Transmitter (UART)

2.3.4.1 TWSI (I2C) Interfaces

The Processor supports the TWSI interface, which can read from and write to devices on an I2C bus. The Processor has two independent TWSI ports.

- TWSI0 connects to the two Mini-DIMM sockets (0 and 1) and is hardwired to device address **0x50** and **0x51** respectively. This TWSI interface is used to determine the characteristics of each installed Mini-RDIMM.
- TWSI1 connects to the FPGA, ZL30152 and two IR3541 devices.

The TWSI interface communicates with any of the cnMIPS cores or a remote host. Software primitives provide access to the TWSI. See “[Chapter 4: Software Description](#)” for more information on TWSI software primitives.

The Processor supports normal and fast modes for the TWSI interface as well as both 7- and 10-bit interfaces. When selected, the Processor and FPGA can operate as *Master* of the TWSI device to initiate a transaction.

Typically, TWSI devices on the WANic-66512 have offset address bits that come out to physical pins so they can use the addressing scheme shown Figure 2-3. Reading from a Most Significant Bit (MSB) and ignoring the Least Significant Bit (LSB) provides the Physical Address. Essentially, this puts a 0x50 offset to the hard strapped address bits A_2 , A_1 , and A_0 on the WANic-66512.

Adding the *Offset* to the hard set *Base Address* results in the *Physical Address* of each device on the WANic-66512 as shown in Table 2-2. However, when the FPGA is the master, each of those devices is located on its own bus, as indicated by the Bus Name column. There is a local FPGA addressing scheme to get to those busses.



NOTE

The connection of the FPGA to the Processor on TWS1_ is not used by any logic; those pins are permanently tri-stated on the FPGA. However, this is provided for the benefit of boundary scan programming and to allow access to the IR3541, and in system programming.

Figure 2-3 TWSI Addressing Format

Device Address							
1	0	1	0	A_2	A_1	A_0	RW
MSB							LSB

Table 2-2 shows the high-level addresses for the TWSI devices on the WANic-66512 and identifies their master controller.

Table 2-2 Local TWSI/I2C Device Addresses

Master Controller	Bus Name	TWSI Device	Base Address	Offset	Physical Address	Local FPGA Address
Processor	TWS_	DIMM0	0x50	0x00	0x50	—
Processor	TWS_	DIMM1	0x50	0x01	0x51	—
Processor	TWSI_	FPGA	—	—	—	—
Processor	TWSI_	ZL30152 SyncE Device	0x28	0x00	0x28	—
Processor	TWSI_	IR3541 (1)	0x0D	0x02	0x0F	—
Processor	TWSI_	IR3541 (2)	0x0D	0x04	0x11	—
FPGA	E2_	EEPROM	0x50	0x00	0x50	0x52
FPGA	SFP_A_I2C	SFP0	0x50	0x00	0x50	0x53
FPGA	SFP_B_I2C	SFP1	0x50	0x00	0x50	0x54
FPGA	TEMP_	Temperature Sensor	0x2C	0x00	0x2C	0x57

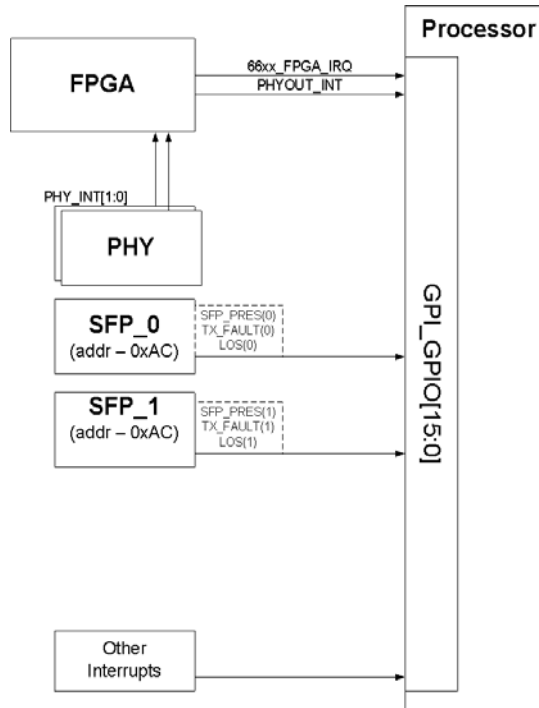
A dash (—) in this table indicates that the device does not have an address.

2.3.4.2 GPIO Interface

The Processor provides a 16-Bit GPIO general purpose interface, which is used as an input port to detect the following signals:

- SFP presence (SFP_PRES)
- Transmit fault (TX_FAULT)
- SFP Loss of Signal (SFP LOS)

Figure 2-4 GPIO Interface Assignment



The GPIO pins can act as either:

- Input — to read digital signals from a circuit; that is, when configured for input mode, they read the current state as either high or low.
- Output — to control or signal other attached devices that the pins are configured to be high or low.

Each bit within the GPIO interface can be configured independently to generate an interrupt in response to a state change for the specific device as described in Table 2-3.

Table 2-3 GPIO Interface Bits

Pin	Field	Description	Value	Input/ Output
GPIO17	NC	Not connected		
GPIO16	NC	Not connected		
GPIO15	GPIO_CLK_0	GPIO Primary Clock 0 – Provides the primary timing reference for the SyncE clock generator from one of the two 10 GbE interfaces.		Output
GPIO14	Interrupt – Thermal Event	Thermal Interrupt Event – When set to 1, indicates an interrupt from the Temperature Sensor to the FPGA, and serves as an overall multiplexed interrupt for Thermal and FPGA events. [See Section 3.1.1 "Temperature Sensor Interrupts" for more information.]	0 = No interrupt 1 = Interrupt	Input
GPIO13	PHY_MUXED_INT	PHY Multiplexed Interrupt from FPGA – When set to 1, indicates multiplexed/masked PHY interrupt from the FPGA.	0 = No interrupt 1 = Interrupt	Input
GPIO12	~DIMM_SLOT_1_TEVENT	DIMM Slot 1 Thermal Event – When set to 1, indicates a Thermal Event occurred on DIMM Slot 1.	0 = Normal 1 = Thermal Event	Input
GPIO11	GPIO_CLK_1	GPIO Secondary Clock 1 – Provides the secondary timing reference for the SyncE clock generator from one of the two 10 GbE interfaces.		Output
GPIO10	~DIMM_SLOT_2_TEVENT	DIMM Slot 2 Thermal Event – When set to 1, indicates a Thermal Event occurred on DIMM Slot 2.	0 = Normal 1 = Event	Input
GPIO9	SFP_LOS(B)	SFP Loss of Signal – When set to 1, indicates a loss of signal on SFP B.	0 = No LOS 1 = LOS on SFP B	Input
GPIO8	SFP_LOS(A)	SFP Loss of Signal – When set to 1, indicates a loss of signal on SFP A.	0 = No LOS 1 = LOS on SFP A	Input
GPIO7	SYSCLK_GPIO_A	User-defined interrupt from SyncE device (ZL30152).		Input
GPIO6	SYSCLK_GPIO_B	User-defined interrupt from SyncE device (ZL30152).		Input
GPIO5	TX_FAULT(B)	Transmit Fault – When set to 1, indicates a transmit fault on SFP B.	0 = No Fault 1 = Fault on SFP B	Input
GPIO4	TX_FAULT(A)	Transmit Fault – When set to 1, indicates a transmit fault on SFP A.	0 = No Fault 1 = Fault on SFP A	Input
GPIO3	SYSCLK_GPIO_C	User-defined interrupt from SyncE device (ZL20152).		Input
GPIO2	SYSCLK_GPIO_D	User-defined interrupt from SyncE device (ZL20152).		Input
GPIO1	SFP_PRES(B)	SFP Presence – When set to 1, detects presence on SFP B.	0 = Not Present 1 = Present	Input
GPIO0	SFP_PRES(A)	SFP Presence – When set to 1, detects presence on SFP A.	0 = Not Present 1 = Present	Input

2.3.4.3 RS-232 Serial Port Interfaces

The Processor provides interfaces to two general purpose asynchronous communications controllers. These industry standard 16550-style Universal Asynchronous Receiver Transmitters (UARTs) are capable of sending and receiving data at rates up to 10 Mega (M) baud. The Processor also provides a Baud Rate Generator, which drives the baud rate by dividing the Processor core clock frequency by a user-specified 16-bit divisor, multiplied by 16.

UART Port 0 is available for debug purposes via an *optional* Serial Debug Adapter (SDA) on the WANic-66512. (See [Section 2.10.1 J5 Header](#).) The serial port is used for console access and has a baud rate programmable up to 115K.



NOTE

In some software routines, UART 0 supports an operator console port.

2.3.5 Watchdog Timers

There is a Watchdog Timer for each of the 10 cores of the Processor. The Watchdog Timers can be accessed by a core or external PCIe device; however, a Watchdog is typically used with its associate core only.

To configure a Watchdog Timer for a specific core, set the **CIU_WDOGn_MODE** register bits to one of the following:

- *Off* - Watchdog disabled.
- *Interrupt Only* - An interrupt is generated whenever a Watchdog timeout occurs.
- *Interrupt and NMI* - An interrupt and a Non-Maskable Interrupt (NMI) is generated for the core on which the Watchdog timeout occurs.
- *Interrupt, NMI, and Soft Reset* - An interrupt, an NMI pulse, and a soft reset across the entire Processor is generated when a Watchdog timeout occurs.

The Watchdog Timer starts with a value of **CIU_WDOGn_LEN** << 8, and decrement every 256 cycles until it is 'Poked.' The Watchdog Timer for each core can be reset by writing to its respective **CIU_PP_POKE_n** register.

See the *OCTEON II CN66XX Hardware Reference Manual* from Cavium Inc. for additional information for configuring and for integrating Watchdog Timers.

The **CIU_WDOG_n** and the **CIU_PP_POKE_n** register descriptions follow.

2.3.5.1 CIU_WDOG n Register

The **CIU_WDOG n Register** is a 64-bit read/write register that allows configuration of the OCTEON Watchdog Timer (where $n = 0-5$).

CIU_WDOG m Register Address:

```

CIU_WDOG0  0x0001070000000500
.
.
.
CIU_WDOG5  0x0001070000000528
    
```

Table 2-4 CIU_WDOG{0:5} Register Description

Bit	Field	Description	Value	Access
63-46	RSVD	Reserved		
45	GSTOPEN	Global-Stop Enable.	0 = Disable 1 = Enable	R/W
44	DSTOP	Debug-Stop Enable.	0 = Disable 1 = Enable	R/W
43-20	CNT	Count — Number of 256 intervals until the next Watchdog Timeout. This bit is cleared by writing to the CIU_PP_POKEn Register .	<number> = ≤ 256	R/W
19-4	LEN	Length — Watchdog Timer expirations length. The most significant 16 bits of a 24-bit length that decrements every 256 clock cycles.	<length>	R/W
3-2	STATE	State — Number of Watchdog Timeouts since the last Watchdog poke. This bit is cleared by writing to the CIU_PP_POKEn Register .	<number>	R/W
1-0	MODE 0	Mode — Selection that indicates the action to generate on a Watchdog Timeout.	1 - Interrupt only 2 - Interrupt and NMI 3 - Interrupt, NMI, and Soft reset	R/W

2.3.5.2 CIU_PP_POKE n Register

The **CIU_PP_POKE n Register** is a 64-bit read/write register that is used to clear the Watchdog Timer (where $n = 0-5$).

CIU_PP_POKE n Register Address:

```

CIU_PP_POKE0  0x0001070000000580
.
.
.
CIU_PP_POKE5  0x00010700000005A8
    
```

Table 2-5 CIU_PP_POKE{0:5} Register Description

Bit	Field	Description	Access
63-0	RSVD	Reserved – Writes to this register generate the following actions: <ul style="list-style-type: none"> Clears pending interrupts generated by the Watchdog Resets CIU_WDOGn_STATE Sets CIU_WDOG_CNT back to CIU_WDOG_LEN << 8. 	R/W

2.3.6 Reference Clock

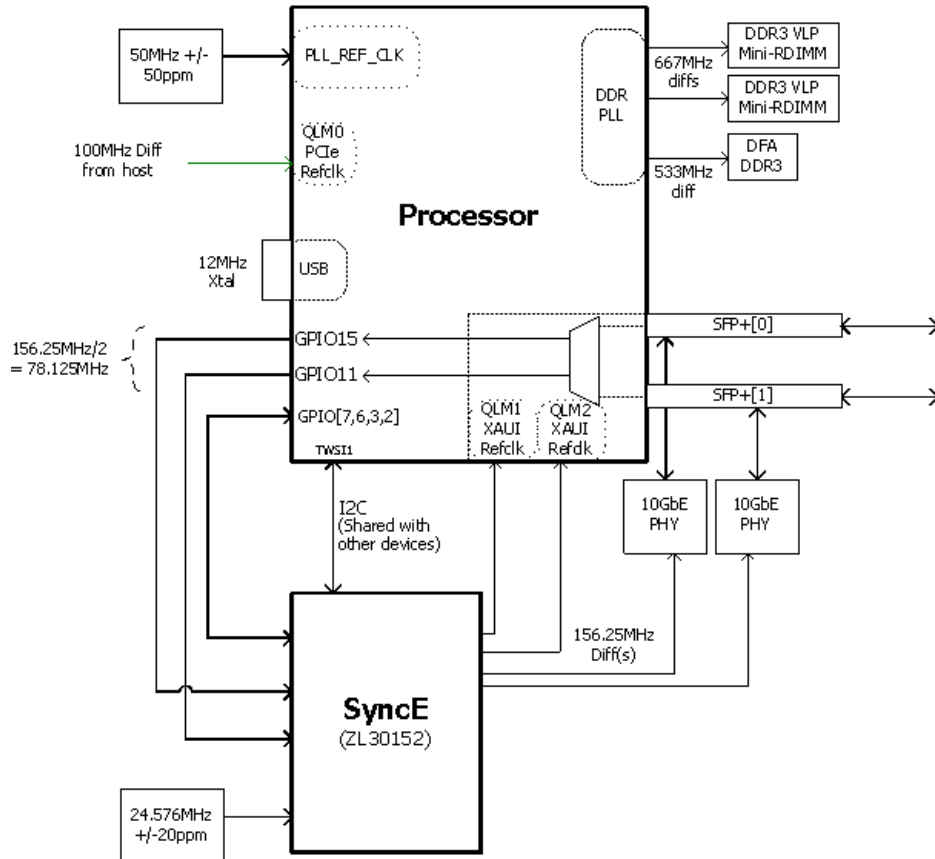
The CN6645 core reference clock input (PLL_REF_CLK) is driven by a fixed 50 MHz oscillator. A Phase Locked Loop (PLL) within the CN6645 multiplies the reference clock to derive the internal core clock frequency according to the 5-bit value strapped to the PLL_MUX_<4:0> OCTEON inputs. The default for the WANic-66512 is a core frequency of 1.5GHz.

The PLL_REF_CLK meets the following clock requirements:

- 50 MHz frequency
- $150 \pm$ total parts per million (ppm)
- 45 to 60% duty cycle
- 150 ps phase jitter (Peak-to-Peak)
- 2.0 ns maximum edge rate (measured from 10–90% single levels on single-edged clock. Differential clock is measured from –150mV to +150mV)
- 0 – 3.3 V levels

Figure 2-5 illustrates the clocking for the WANic-66512.

Figure 2-5 WANic-66512 Clocking



2.4 Synchronous Ethernet Clock (SyncE)

SyncE is implemented with Zarlink Semiconductor's ZL30152 device to ensure that the Processor always derives timing from the most reliable source. The SyncE device delivers hitless switching (*uninterrupted*) between timing references, and makes transmitting out of the Processor synchronous to any line interface it receives.

The SyncE device selects the clocks from the two SFP+ devices:

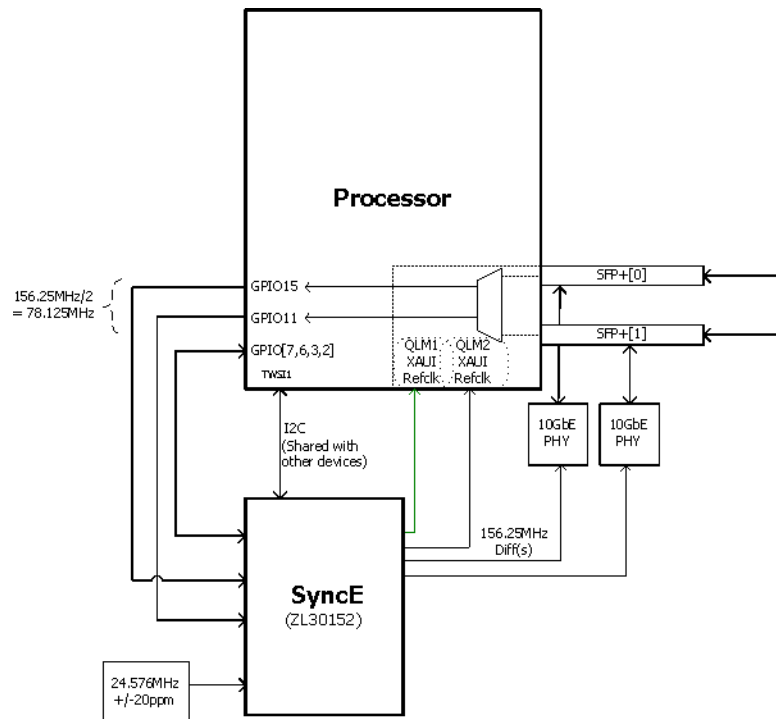
- One SFP+ device is selected as a *Primary Timing Source*.
- A second SFP+ device provides a *Secondary Timing Source*.

These two clocking sources are routed out the Processor on **GPIO15** and **GPIO11** as described in Table 2-3 and shown in Figure 2-6.

The SyncE device also uses these two selected clocks as inputs in addition to a 20ppm local oscillator. This SyncE device determines which of these three clocks is used at any one time for the output clock from the SyncE device, which is fed back to the CN6645 as the timing source for both XAUI QLM block as shown in Figure 2-6.

Four configurable GPIO pins (2, 3, 6, and 7) are wired from the SyncE device to the Processor. See [Section 2.3.4.2 "GPIO Interface"](#) for additional information on these GPIO pins.

Figure 2-6 Synchronous Ethernet Clock



NOTE from Cavium:

"There is a reference clock to the QLM and we phase align it with the input clock from the SerDes lane to give you the GPIO output clock. Hence, when the link is removed, the clock keeps running."

2.5 FPGA

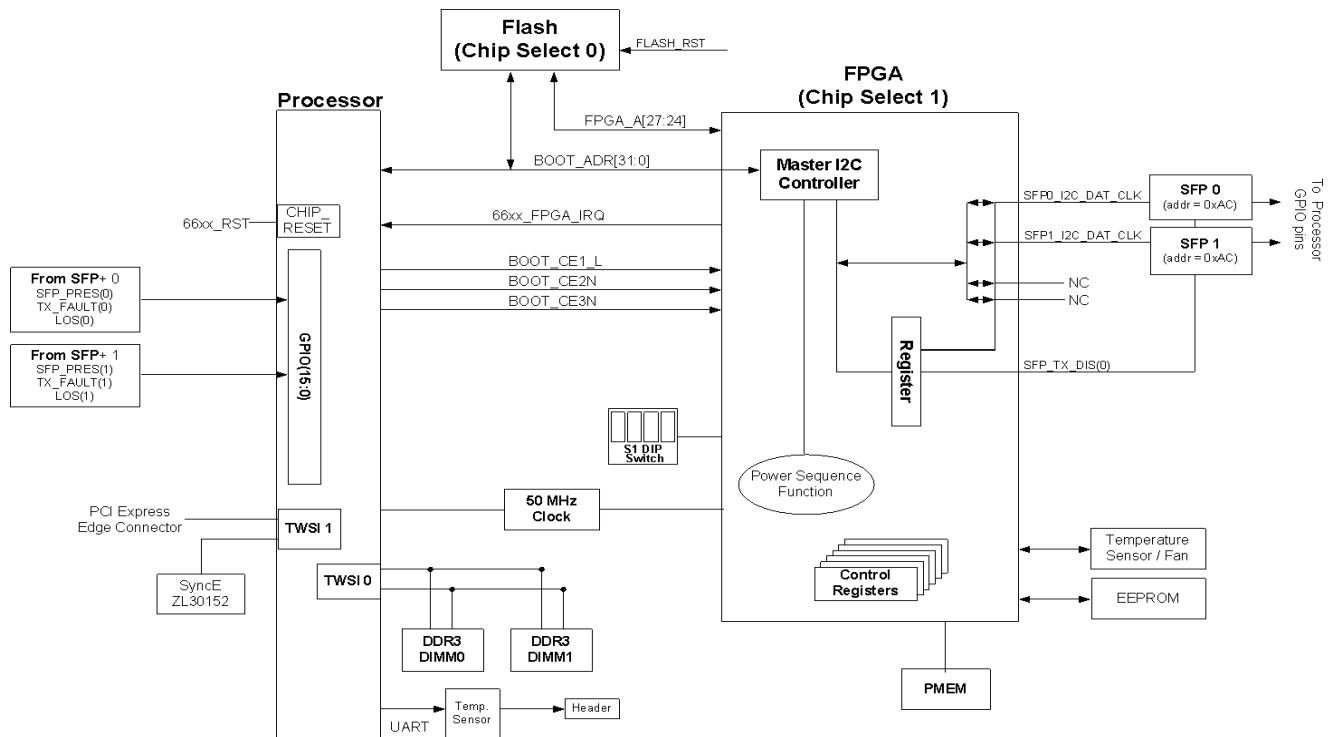
The WANic-66512 use an FPGA as the interface to various on-board hardware resources. The FPGA resides on the Processor Boot Bus (along with the boot Flash device) and can be accessed by the cnMIPS cores or PCIe Host.

The FPGA includes a single master I2C controller to access the management port of each SFP+ module, and the power sequencing function.

The FPGA provides the following functions and interfaces as shown in Figure 2-7:

- Control and Status Registers
- PMEM
- 50 MHz Clock
- Interrupt Control
- I2C Interface for SFP+ and EEPROM
- FPGA registers accessible by Boot Bus
- Power-on Reset
- On-Board Thermal Protection

Figure 2-7 FPGA Function Block Diagram



2.5.1 Control and Status Registers

The FPGA contains special registers that allow the Processor to control and to monitor transactions among interfaces and local hardware resources. Boot Chip Select 1 decodes access to various control and status registers in the FPGA. The FPGA latches the upper three address bits for the Boot Flash using the Boot Bus. A dedicated register, **Boot Flash Upper Address Control Register**, is provided for this operation.

All FPGA registers map to a contiguous block of 16 bytes within each address space. Thus, the FPGA Chip Selects can be configured to locate this bank of 32 locations to any area within each address space. In cases when the Chip Select logic configuration accesses a range of addresses larger than 16 bytes, the FPGA register image repeats.

2.5.2 Persistent Memory (PMEM)

The WANic-66512 contains PMEM as non-volatile storage. The contents of PMEM is preserved as long as power is applied. PEM is further described in [Section 2.6.4 Persistent Memory \(PMEM\)](#).

The FPGA has a dedicated interface to access PMEM. To access PMEM from the Boot Bus, Chip Select 2 (BOOT_CE2N) and Chip Select 3 (BOOT_CE3N) are brought to the FPGA. Use BOOT_CE2N for the lower 16MB of PMEM, and BOOT_CE3N for the upper 16MB of PMEM.

2.5.3 FPGA Clocking

As described in [Section 2.3.6 Reference Clock](#), the FPGA uses a copy of the 50 MHz Processor core reference clock as its primary synchronization method.

2.5.4 FPGA Interrupts

The FPGA provides a means to generate an interrupt upon completion of a transaction on any of its I2C interface ports. Generation of interrupts can be enabled on a port-by-port basis by means of bits in the FPGA **Interrupt Control Register**. (See [Chapter 3: FPGA Registers](#) for more information on this register.)

2.5.5 FPGA Reset Control

The FPGA routes individual reset signals to the:

- Flash memory
- DIMM memory
- 10Gb PHY device

These resets signals are asserted at power-up and are de-asserted automatically when power is stable. Software can force an individual reset to these devices using the **Peripheral Reset Register**. (See [Chapter 3: FPGA Registers](#) for more information on this register.)



CAUTION

A PCI reset from the Host does **not** reset these devices.

2.5.6 I2C Interface

The FPGA provides independent I2C port interfaces for the following devices:

- Front panel SFP+ devices
- 64 KB serial EEPROM
- MAXIM® 6663 temperature sensor

The FPGA performs read and write operations on each I2C interface in response to requests from the Processor.

An independent set of registers is provided for each I2C interface. See “[Chapter 3: FPGA Registers](#)” for a description of each I2C interface register.

2.5.7 TWSI Interface

The FPGA can be master controller for the TWSI devices listed in Table 2-6.

Table 2-6 FPGA TWSI Devices and Addresses

Device	Local FPGA Address
EEPROM	0x52
SFP0+ (top)	0x53
SFP1+ (bottom)	0x54
Temperature Sensor	0x57

A description of each device is given in this chapter.

2.5.8 FPGA Power Supply Control

The FPGA enables various on-card power supplies to turn on in a specific sequence with controlled timing. This operation is automatic upon application of +3.3V and +12V power to the input of the WANic-66512.

The FPGA also monitors the output state of each individual power supply during the power-up sequence, and does not advance to the next step in the sequence until the previously enabled supply is also verified to be within the specified operating range.

As an added safeguard, when any of the supplies fail to reach the specified output voltage range within a prescribed timeout period (indicating that a possible overload condition exists), all supplies are shut down indefinitely until the input power to the card is removed and re-applied.

2.5.9 On-Board Thermal Protection

On-Board Thermal Protection is designed to protect the WANic-66512 from damage due to overheating caused by inadequate airflow to the board. When the WANic-66512 receives consistent airflow and cooling, as described in [Section 1.3 Air Flow and Cooling Requirements](#), this condition does not occur.

On-Board Thermal Protection powers down the WANic-66512 automatically when the Temperature Sensor indicates that the Processor die temperature or board's ambient temperature has exceeded the safe operating range. All on-board supplies are disabled and the WANic-66512 ceases operation; therefore, all packet processing is terminated and the Power LED turns *RED*. The user must perform a system airflow evaluation to correct all airflow problems. When the required airflow and cooling environment is established, power-cycle the system to recover from this condition.

When an overtemperature event is detected by the Temperature Sensor, an interrupt is sent from the Temperature Sensor through the FPGA so the interrupt may be masked when necessary. The interrupt is run from the FPGA to the **GPIO14** pin on the Processor. This interrupt toggles before the shutdown signals out of the Temperature Sensor allowing software to take any required action for the thermal event. If the `shutdown_remote` or `shutdown_local` interrupt is seen at the FPGA, there is a 10 millisecond (ms) window for the interrupt to clear; otherwise, 20ms after the thermal event, the FPGA shuts down power supplies leaving only the 3.3V power supply active.

2.6 Memory

The packet memory on the WANic-66512 consists of external SDRAM. Flash memory provides storage for start-up boot code and Processor core images. The EEPROM provides general purpose storage of hardware information. The WANic-66512 also contains Persistent Memory and an embedded USB Flash disk drive.

2.6.1 Internal (Level 2) Cache Memory

The Processor utilizes internal Level 2 cache by means of a coherent memory bus. The Level 2 cache connects to two DDR3 controllers to support DDR3 SDRAM packet memory, which provides primary data storage.

2.6.2 DDR3 SDRAM

The WANic-66512 supports DDR3 memory with a 64-bit data path. An additional 8 bits is provided for Error Correction Code (ECC) support. ECC permits detection and correction of single-bit memory errors as well as two-bit error detection. Timing for the DDR3 SDRAM interface is derived from an internal Phase Lock Loop (PLL) device that locks to and multiplies from the core clock frequency.

A power subsystem (IR Subsystem) provides voltage tracking and over-voltage protection. A special register, **IR3541 Interrupt Register**, indicates when the IR Subsystem detects an alarm condition or an over-temperature condition for the DDR3 memory (or the Processor). See “[Chapter 3: FPGA Registers](#)” for a description of this register.

Mini-RDIMMs

The WANic-66512 uses Very Low Profile (VLP) Mini-RDIMMs with ECC and supports both single- and dual-rank varieties. The Mini-RDIMMs connect to the card by means of two angled VLP Mini-DIMM sockets.

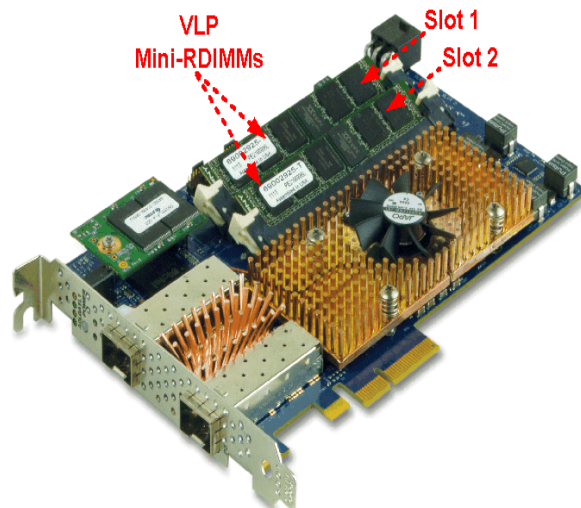
Mini-RDIMM sockets (Slot 1 and Slot 2) are connected to the Processor, TWSI, I2C interface, and hardwired to device addresses **0x50** and **0x54** respectively. Figure 2-8 shows the location of Mini-RDIMMs and slots on the WANic-66512.



NOTE

Single RDIMM use is *only* supported in Slot 1, which is furthest from the OCTEON Processor.

Figure 2-8 Mini-RDIMMs on WANic-66512



Serial Presence Detect

Each VLP Mini-RDIMM socket features Serial Presence Detection (SPD) based on a serial EEPROM device on the Mini-RDIMM, which uses the I2C protocol to access information about each Mini-RDIMM.

Reset

Mini-RDIMM reset is asserted at power-up and de-asserted automatically once power is stable. Software can force an individual reset to the Mini-RDIMM using the **Peripheral Reset Control Register**. See “[Chapter 3: FPGA Registers](#)” for a description of this register.



CAUTION

A `PCI_RESET` signal from the Host does **not** reset the Mini-RDIMMs.

HFA/DFA Engine

The HFA/DFA Engine is the Processor string and pattern matching regular expression accelerator engine, which is supported by a dedicated DDR3 memory controller and external DDR3 memory. Up to 512 MB of DDR3 memory is provided to support the HFA/DFA engine.

2.6.3 Boot Flash

An on-board Flash memory device operates as a 128 MB device organized as 1024 blocks of 128 Kb each. Code execution is supported from this Flash during hardware initialization *only*. The Flash is directly connected to the Boot Bus.

Operating firmware copies the boot code from Flash into the DDR3 SDRAM packet memory for use (following hardware initialization) to optimize code performance. When required by operating firmware, the Flash device can be written to or erased.

The Flash is partitioned at the factory and can be reprogrammed. For information on reprogramming the Flash, as well as information on accessing and partition the Flash, see “[Section 4.5.3 Flash Driver](#).”



NOTE

The WANic-66512 also supports PCIe booting. See “[Section 2.11 “Dual In-Line Package \(DIP\) Switch”](#)” for more information on selecting PCIe booting.

2.6.4 Persistent Memory (PMEM)

PMEM is a storage device whose contents are preserved when the power is On. The WANic-66512 provides 32 MB of PMEM, which retains its contents across hard and soft resets *except* for power-on resets. PMEM is available to store system events and logs messages for use after a system failure. The contents of PMEM is preserved over any system reset, and is available for analysis after a system failure and reboot.



NOTE

The PMEM contents **is not preserved** whenever power is not applied (for example, if there is no battery backup.)

The 32 MB of shared memory is treated as two 16 MB regions. Each region is accessed through memory Chip Select 2 (`BOOT_CE2N`) to provide the lower PMEM address and Chip Select 3 (`BOOT_CE3N`) provides the upper PMEM address.

2.6.5 Serial EEPROM

The Serial EEPROM provides 64 KB of general-purpose non-volatile data storage for on-board specific hardware configuration information, such as assembly model number, serial number, and so forth. An I2C interface connects the EEPROM to the Processor by means of the FPGA. This permits examination and modification of the EEPROM contents.

2.6.6 USB Flash Drive

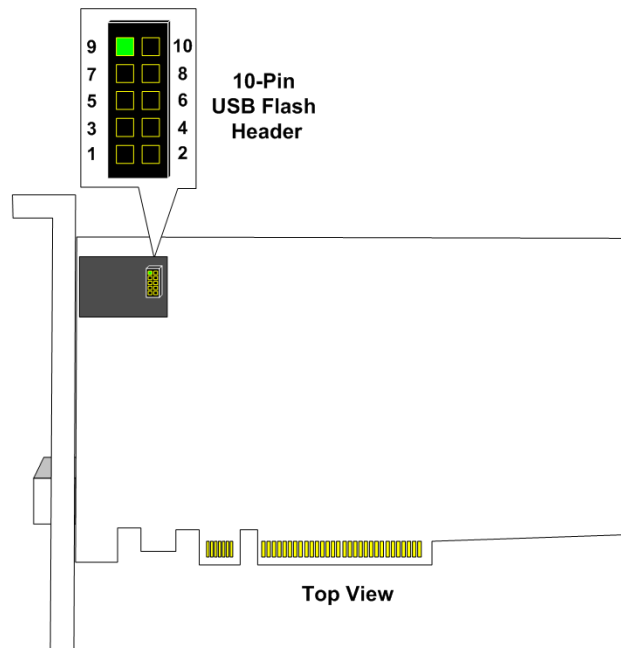
The WANic-66512 contains a removable 2 GB non-volatile, solid-state USB Flash drive for additional storage or application usage. The USB Flash drive contains a standard 10-pin header on the top (circuit side) of the WANic-66512, similar to that shown in Figure 2-9.



NOTE

For additional USB configurations of up to 4 GB, consult a GE Intelligent Platforms Customer Support Representative.

Figure 2-9 USB Flash Drive Header



The pinout for the USB Flash drive header is described in Table 2-7.

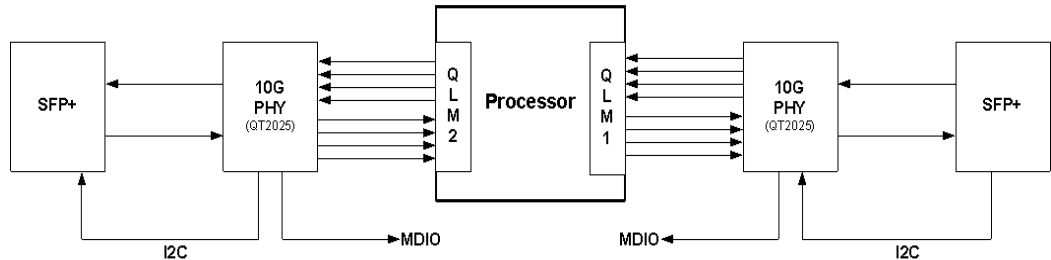
Table 2-7 USB Flash Header Pinout

Pins	Type	Signal Name	Description	Illustration
1	Power	VBUS	Bus voltage supply from source(+5V)	
3	D-	i/O	Data line -	
5	D+	I/O	Data line +	
7	GND	Ground	Ground	
2, 4, 6, 8, 10	NC	Open	No connect	
9	Key	Open	Alignment	

2.7 Gigabit Ethernet Physical Device

The 10G Physical device (PHY) is a QT2025 from AppliedMicro™, which provides a high-performance 10 Gbps transceiver independent port. As shown in Figure 2-10, the 10G PHY device connects to a XAUI link on the Processor through QLM1 and QLM2. The 10G PHY device implements the IEEE 802.3u Physical Coding Sublayer (PCS) Clause 22 specification for 10GBASE-SR and 10GBASE-LR.

Figure 2-10 10G PHY Implementation



PHY A maps to Port 0, and PHY B maps to Port 1.

The 10G PHY device reset is asserted at power-up by the FPGA, and de-asserted automatically when power is stable. Software can force an individual reset to the PHY device using the **Peripheral Reset Control Register**.



A PCI_RESET from the host device does **not** reset this device.

Interrupts on each PHY are routed to the FPGA. The **PHY Interrupts Register** allows the masking or activation of a PHY interrupt. See [Chapter 3 • "FPGA Registers"](#) for more information on register descriptions.

2.8 Power Distribution

The power distribution system has multiple power sources; however, there are two main supplies:

- +3.3V — comes from the PCIe edge connector.
- +12V — configured to pull from an external power connector

In addition, a voltage regulator subsystem provides voltage tracking and over-voltage protection.

2.8.1 PCIe Edge Power

Table 2-8 lists the PCIe slot voltage limits as described in the PCIe V2.1 specification.

Table 2-8 PCIe Slot Power Limits

Supply (Volts)	Maximum Operating Current (Amps)	Maximum Power (Watt)
+3.3V	3.0A	10W
+12V	2.1A	25.2W

In addition, a +12V PCIe Graphic Power Connector is provided via an external power connector. (See [Section 1.4 "Power"](#) and [Table 1-4, "WANic-66512 Current and Power Limits"](#) for additional information.)

PCIe Edge Fingers

The PCIe Edge Fingers provides a reliable, high-speed, serial point-to-point interconnect that is compatible with the PCIe Version 1.x and V2.x buses. The PCIe Edge Fingers support speeds up to 2.5 Gb/s per lane in each direction for PCIe V1.1, or up to 5.0 Gb/s for PCIe V2.0. The PCIe Edge Fingers achieve reliable, high data rates by using Low Voltage Differential Signaling (LVDS) pairs. The PCIe Edge Fingers provide a four lane PCIe interface.

The WANic-66512 uses the +3.3V from the PCIe Edge Fingers in all board configuration. This +3.3V supply also creates the core voltage (+1.5V) for the FPGA and starts to activate the other supplies in a serial fashion.

Clocking Specifications

The PCIe Edge Fingers must be supplied with a clock from the Host computer, which must meet the following clock requirements:

- 100 MHz frequency
- 300 ± total ppm (part per million)
- 40 to 60% duty cycle (active time)
- 840ps edge rate (measured from 20–80% single levels on single -edged clock. Differential clock is measured from -150mV to +150mV)
- 0 – 0.7 V nominal levels
- 100 Ohm source termination
- All jitter requirements per PCIe specifications

2.8.2 IR Subsystem

The IR Subsystem is comprised of an IR3541 Multiphase Controller and an IR3550 Integrated Power Phase device from International Rectifier®.

The IR3541 is a flexible, dual-loop, digital multiphase buck (step-down) controller optimized to convert a +12V input supply to core voltage for the Processors and DDR memory. The IR3541 device is programmed with configuration files at boundary scan step for each board and loads customized settings into non-volatile memory (NVM.)

The IR3550 is a synchronous buck gate driver integrated circuit with co-packed control and synchronous Metal–Oxide–Semiconductor Field-Effect Transistor (MOSFETs) and Schottkey diode. It is optimized internally for heat transfer and driver/MOSFET timing and offers superior performance by achieving improved electrical efficiency to help reduce overall power losses and heat generation in the system, thereby improving reliability.

The IR Subsystem delivers the following voltage:

- Vcore — Processor core voltage
- Vmem — Memory voltage
- Vtt — Voltage transient tool

Use the FPGA **IR3541 Status Register** to display the status of the IR3541 Subsystem.

2.9 Temperature Management

The WANic-66512 contain a MAXIM™ 6653 combination fan controller and temperature sensor to actively monitor the temperature of the Processor as well as WANic-66512 local temperature. The Temperature Sensor provides *set points* (maximums) for the temperatures (CRIT0, CRIT1). When a set point is crossed, the Temperature Sensor asserts the thermal overload signal (~THERM). In addition, two shutdown outputs ~SHUTDOWN-Remote, ~SHUTDOWN-Local) are triggered when the remote or local temperatures exceed the programmed set points.



NOTE

Remote and Local are in the perspective of the Temperature Sensor. For example, Remote is the CN6645 thermal diode, and Local is the Temperature Sensor itself.

The FPGA uses the CRIT0, CRIT1, ~THERM, and ~SHUTDOWN signals on the MAXIM 6653. The FPGA has the ability to monitor Shutdown alarm temperatures based on thresholds set by the combination of the CRIT1 and CRIT0 signals that are set by hardware straps.

CRIT1 and CRIT0 must be set at device power-up. However, all temperature threshold limits are stored in the threshold limit registers and can be changed through the SMBus digital interface.

- When the ~THERM set points are tripped, the FPGA is alerted by the ~INT signal.
- When Shutdown-Remote is tripped, the FPGA is notified by a low on the ~SDR (~Shutdown_Remote) signal.
- When Shutdown-Local is tripped, the FPGA is notified by a low on the ~SDL (~Shutdown_Local) signal.

Also, the FPGA provides two status registers to indicate to the Host when these set points are exceeded:

- The **Interrupt Status Register** indicates to the Host that there was an interrupt involving the Temperature Monitor.
- The **Temperature Status Register** provides information to the Host as to the cause of the interrupt.

See [Chapter 3: FPGA Registers](#) for more information on these FPGA registers.

2.9.1 Monitoring Temperature from the Host

The WANic-66512 and the OCTEON core temperatures can be monitored from the Host using the /proc file. The /proc file is created and updated by the NPA Driver. The /proc file provides access to hardware and firmware information, and allows the Host to analyze the WANic-66512 temperature conditions as part of the WANic-66512 health check.

After installing the NPA Driver, enter the following command to display the temperature readings:

```
/proc/drivers/gefes<x>/temperature
```

where <x> = device number

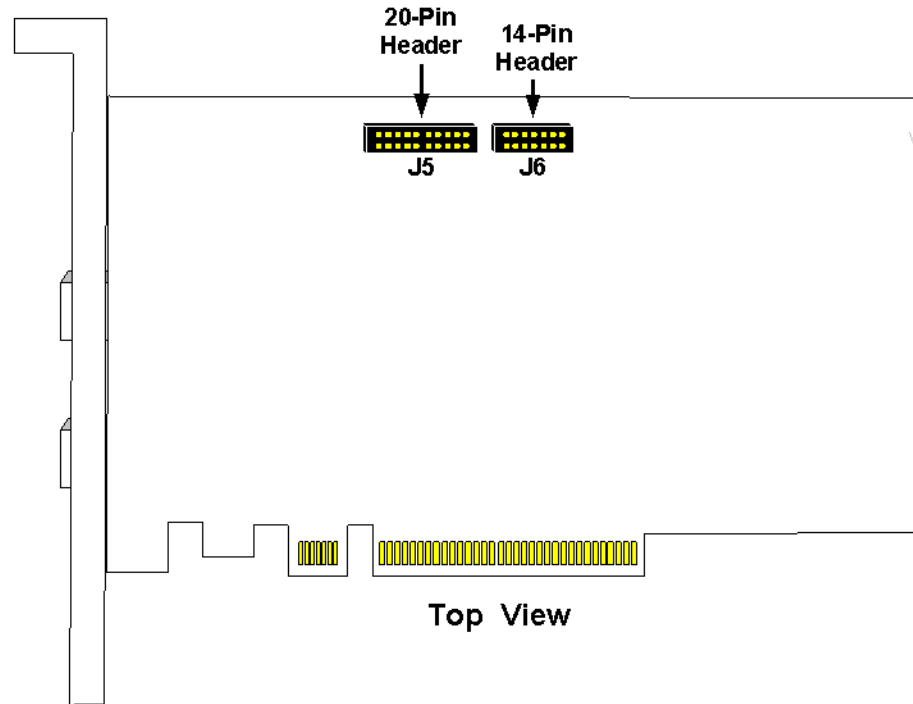
Refer to [Chapter 4 • "Software Description"](#) for more information on the NPA Driver and the /proc file system.

2.10 Headers

The WANic-66512 contains J5 and J6 featured headers which are located on the top of the board (component side) similar to that shown in Figure 2-11.

- J5 – is a 20-pin header for access to RS-232 and External Joint Test Action Group (EJTAG) scan chain interfaces
- J6 – is a 14-pin header for the JTAG interfaces

Figure 2-11 Header Locations



2.10.1 J5 Header

The EJTAG and RS-232 UART are combined on a single 20-pin header, J5. Certain signals within the JTAG port are also shared with the EJTAG scan chain. The on-board S1 DIP Switch bank determines whether these signals are connected as part of the scan chain or whether they are routed out to the header.

An RS-232 header provides the signals for one UART depending on the WANic-66512 model.

Serial Debug Adapter (optional)

An *optional* Serial Debug Adapter (SDA) and cable assembly separates the JTAG and RS-232 signals into two independent standard boxed-headers:

- RS-232 – 10 pin connector
- EJTAG – 14-pin connector



NOTE

Contact a GE Intelligent Platforms sales representative for information on purchasing the *optional* SDA and appropriate cables.

The RS-232 10-pin connector permits the attachment of a standard Insulation Displacement Connector (IDC) cable assembly consisting of a 10-pin header and a standard 9-pin RS-232 Digital Communications Equipment (DCE) interface.

The EJTAG 14-pin header supports direct attachment of EJTAG debugging tools. The header is keyed to ensure proper connector alignment when the user-supplied cable assembly is attached to the optional SDA.

Note the following information:

- The pinout and gender of the DB-9 connector permits it to be directly attached to DTE devices such as a dumb terminal or terminal emulation software running on a PC. If an extension cable is required, be sure to use an *RS-232 Straight-Through Cable*. An RS-232 Straight-Through Cable has a DB-9 plug on one end and a DB-9 socket on the other. This cable does not have *'crossover'* or *'null modem'* functionality built into it.
- The RS-232 IDC cable assembly is keyed to ensure proper insertion into the SDA.

2.10.2 J6 Header and JTAG Scan Chain

The J6 Header provides an interface for JTAG testing. To support testing and debugging, all WANic-66512 components that provide JTAG test ports are interconnected to form a JTAG Scan Chain.



CAUTION

To ensure proper operation under normal operating conditions, the JTAG *TRST* signal must be held in its asserted state (*low*). Failure to observe this requirement can result in unpredictable behavior of the module.

Components that connect to the JTAG Scan Chain include the following:

- Processor
- One or two 10GbE PHYs
- SyncE device (ZL30152)
- FPGA

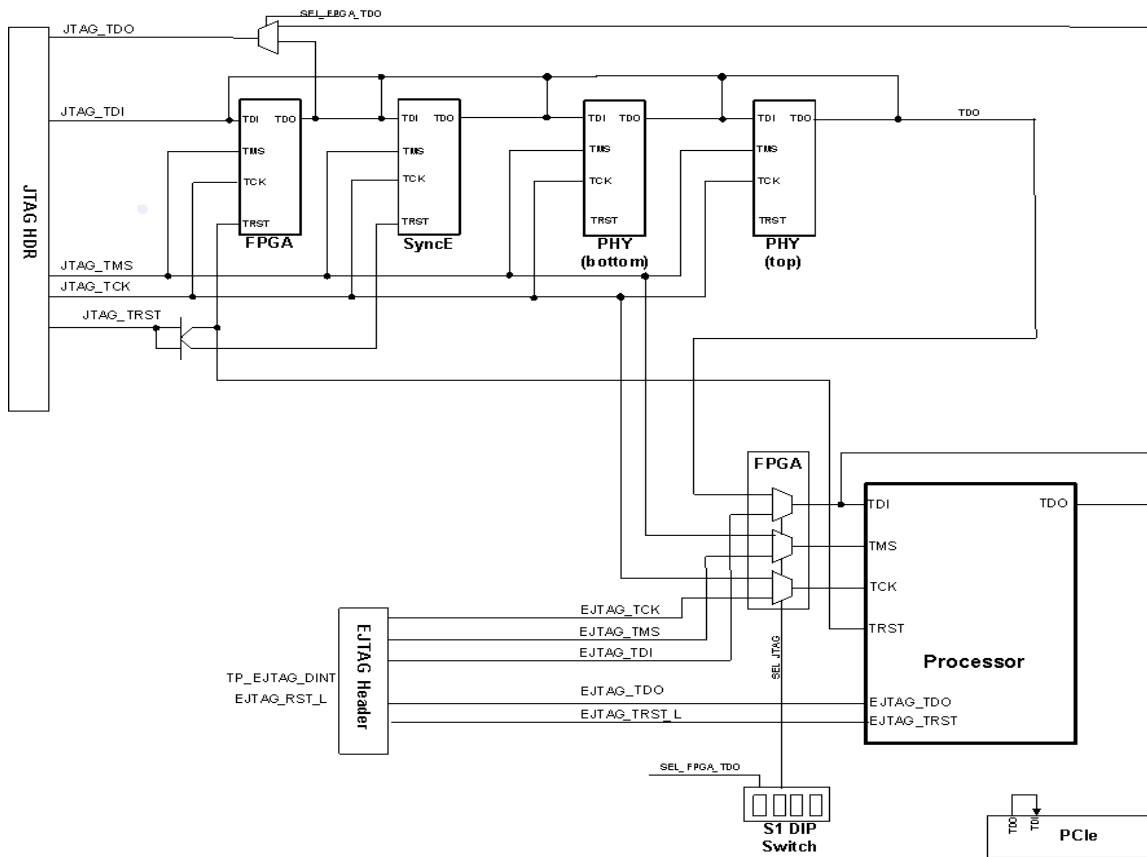
The on-board S1 DIP Switch bank permits the JTAG Scan Chain to be reconfigured to simplify programming of the FPGA, and to support the use of EJTAG debugging tools in conjunction with the Processor. See Figure 2-13 for the location of the S1 DIP Switch bank, which is comprised of four DIP Switches (1-4).

In the S1 DIP Switch bank, set Switch 1 and 2 to the 'ON' position to include all devices in the JTAG Scan Chain. This represents the default configuration for all manufacturing JTAG tests.

When Switch 1 is set to 'OFF', the FPGA Test Data Out (TDO) signal is reconnected directly to the Connector JTAG TDO signal. Since the FPGA Test Data In (TDI) input is the first in the chain, it always connects directly to the TDI input. Thus, setting Switch 1 to 'OFF' effectively isolates all other devices from the JTAG Scan Chain (from the PCIe Connector perspective). Typically, this simplifies the task of in-circuit programming the FPGA with JTAG programming tools.

Setting Switch 2 to the 'OFF' position disconnects certain Processor JTAG I/O signals from the scan chain and reconnects them to the 20-pin header. This configuration permits attachment of an Enhanced JTAG (EJTAG) debugging tool, connecting the device TDI input to the TDO output (normally unpopulated).

Figure 2-12 JTAG Scan Chain



2.10.3 Enhanced JTAG Header (EJTAG)

The Processor JTAG pins can be connected to the JTAG Scan Chain or to the EJTAG header. Multiplexers, under control of the on-board S1 DIP Switch bank, route the JTAG pins.

The EJTAG Header is a standard 14-pin dual-row boxed header connected to the 20-pin header through the SDA card. It provides keying of a mating cable to prevent inadvertent insertion in the wrong direction. The EJTAG Header interface permits the Processor to connect to standard debugging tools during software development. It also provides a means of initially programming a boot-code loader routine into the Flash.

Since the Processor is a target device on the PCIe interface, the `PCI_RST` signal is used as the chip reset for the Processor.

NOTE

The `EJTAG_RST` signal is **not used** in this PCIe target application. The `EJTAG_RST` is not required by the software debugger. When a reset is necessary, the Host must issue a PCI Reset (`PCI_RST`) signal.

2.11 Dual In-Line Package (DIP) Switch

The WANic-66512 contain a grouping of four DIP Switches, collectively labeled S1 as shown in Figure 2-13. Set the S1 DIP Switches to perform the following operations:

- JTAG boundary scan testing
- FPGA override for stand-alone JTAG programming of FPGA
- EJTAG configuration
- Boot mode selection
- Diagnostic utility booting

The **Board ID and DIP Switch Register** permits the Processor to determine the position of each the DIP Switch within the S1 DIP Switch bank.

Figure 2-13 shows the approximate location of the S1 DIP Switch on the bottom (soldered side) of the board. Table 2-9 through Table 2-11 summarize the DIP Switch settings and associated operations.



NOTE

When using switch 3 in the S1 DIP Switch bank to invoke the Flash application boot process, use Environment Variables to control the process. (See [Chapter 4: Software Description](#) for more information on Environment Variables.)

Figure 2-13 S1 DIP Switch Approximate Location

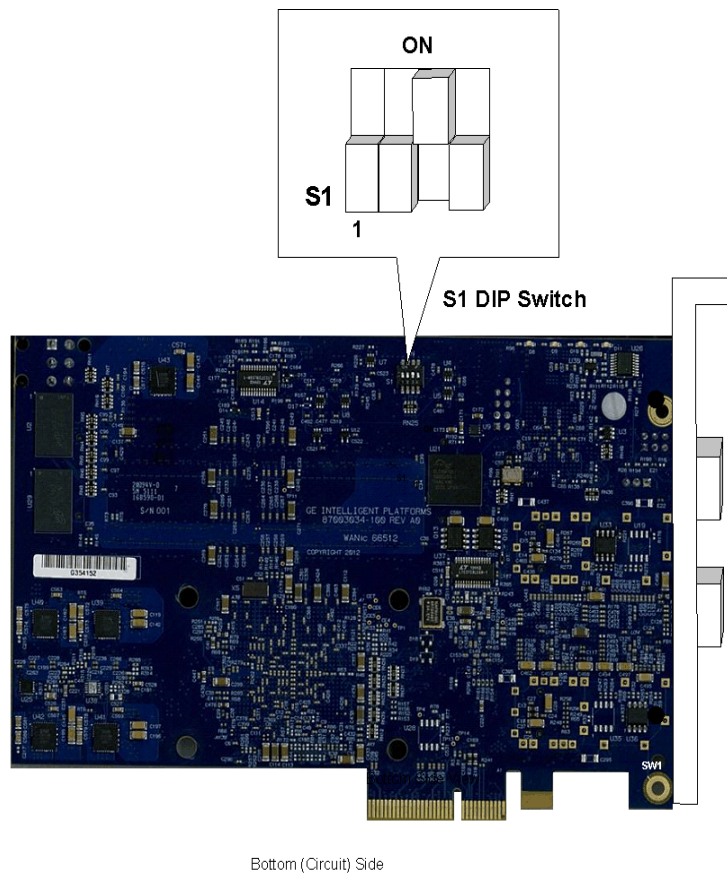


Table 2-9 JTAG Mode S1 DIP Switch Configuration

Mode	Operation	DIP SWITCH	
		1	2
JTAG	Configures the board/MPU for use with EJTAG debugger	Off	On
	Configures the board for JTAG programming of the FPGA	Off	Off
	Not supported	On	Off
	Configures the board for boundary scan testing on full JTAG chain	On	On

Table 2-10 Boot Mode S1 DIP Switch Configuration

Mode	Operation	DIP SWITCH
		3
Boot	Enables Flash Boot operations	Off
	Enables PCIe Boot operations	On



NOTE

When using DIP Switch 3 to invoke the Flash application boot process, use Environment Variables to control the process. (See [Chapter 4: Software Description](#) for more information on Environment Variables.)

Table 2-11 Diagnostic Mode S1 DIP Switch Configuration

Mode	Operation	DIP SWITCH
		4
Diagnostic	Disables Linux auto-start	Off
	Enables auto-start Linux and Diagnostics	On

3 • FPGA Registers

This chapter describes the FPGA registers that the WANic-66512 uses to communicate with other on-board components.

3.1 FPGA Register Summary

The register interface provides direct access to FPGA registers. These 8-bit address registers, listed in Table 3-1, are used for the following:

- Resets
- Interrupts
- LEDs
- Temperature Sensor Control
- Status indications

The base address of the FPGA in U-Boot or Linux is **8001 0000** hexadecimal (h).

Access is R=Read, W=Write, O=Only.

Table 3-1 FPGA Memory Mapped Registers

Offset	Register	Description	Access
0x00	FPGA Revision	Provides the FPGA revision number.	R/O
0x01	Board ID and DIP Switch	Determines hard straps and DIP Switch settings.	R/O
0x02	LED Control	Manages the four diagnostic module status LEDs.	R/W
0x03	Interrupt Control	Controls interrupts for selected devices.	R/W
0x04	Interrupt Status	Indicates interrupt status for selected devices.	R/W
0x05	Peripheral Reset	Generates a reset for selected on-card devices.	R/W
0x06	Transmit Control	Manages the front panel SFP+ transmitter output.	R/W
0x07	Fan and Temperature Status	Indicates when the temperature sensor or fan has experienced a fault.	R/W
0x08	Temperature I2C Control	Indicates an I2C transaction for the temperature monitor.	R/W
0x09	Temperature I2C Address High	Specifies the upper address for the transaction.	R/W
0x0A	Temperature I2C Address Low	Specifies the lower address for the transaction.	R/W
0x0B	Temperature I2C Data	Contains the data for the transaction.	R/W
0x0C	EEPROM I2C Control	Initiates an I2C transaction and, when necessary, reports an error for the EEPROM.	R/W
0x0D	EEPROM I2C Address High	Specifies the upper address for the transaction.	R/W
0x0E	EEPROM I2C Address Low	Specifies the lower address for the transaction.	R/W
0x0F	EEPROM I2C Data	Contains the data for the transaction.	R/W
0x10 – 0x17	Reserved		
0x18	SFP1 I2C Control	Initiates an I2C transaction and reports an error for SFP 1.	R/W
0x19	SFP1 I2C Address High	Specifies the upper address for the transaction.	R/W
0x1A	SFP1 I2C Address Low	Specifies the lower address for the transaction.	R/W
0x1B	SFP1 I2C Data	Contains the data for the transaction.	R/W
0x1C	SFP0 I2C Control	Initiates an I2C transaction and reports an error for SFP 0.	R/W
0x1D	SFP0 I2C Address High	Specifies the upper address for the transaction.	R/W

Table 3-1 FPGA Memory Mapped Registers

Offset	Register	Description	Access
0x1E	SFPO I2C Address Low	Specifies the lower address for the transaction.	R/W
0x1F	SFPO I2C Data	Contains the data for the transaction.	R/W
0x20	Boot Flash Upper Address Control	Breaks the 1 GB Flash into 16 MB pages.	R/W
0x21	Miscellaneous Functions	Provides several miscellaneous functions.	R/W
0x22	PHY Interrupt	Controls interrupts for on-card PHYs.	R/W
0x23	Thermal Interrupt	Provides thermal fault interrupts.	R/W
0x24	IR3541 (Voltage Regulator) Interrupt	Provides interrupts for the IR3541 device.	R/W
0x25	DIMM Thermal Event	Provides interrupts for both DIMMs.	R/W

3.1.1 Temperature Sensor Interrupts

The following registers are grouped in the FPGA to allow the Temperature Sensor direct input into the Processor:

- PHY Interrupts Register (0x22)
- Thermal Interrupt Mask Register (0x23)
- DIMM Thermal Event Register (0x25)

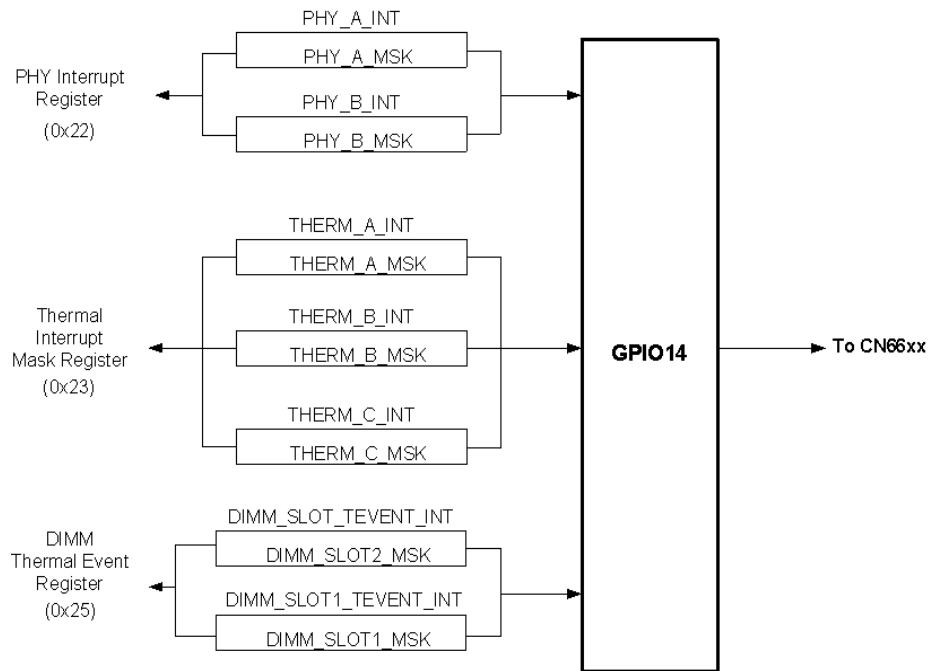
These registers are grouped in the FPGA to a common GPIO (**GPIO14**) as `FPGA_INT`. Their (inverted) state is also registered at each specific register address so software can see the common interrupt at the Processor and read the register to see specifically which thermal fault caused the interrupt. It is possible that more than one thermal fault is interrupting when the register is read.

The read-only bits of these registers report the current state of the interrupt bits directly from the Temperature Sensor. These bits remain active until the issue is resolved at the Temperature Sensor.

The upper 4 bits of the register serve as the masking bits for each of the interrupts. Any combination of an interrupt active and the mask active generates GPIO14 active, as illustrated in Figure 3-1. For example, in the **Thermal Interrupt Mask Register**, both `THERM_A_MSK` must be active (1) and the input `THERM_A_INTN` must be active ('1') for GPIO14 to the Processor to be active ('1').

The interrupt generated by the Temperature Sensor is a current indication of its status. As soon as the interrupt is serviced, the signal is de-asserted. Therefore, the lower 4 bits of the register updates the current state of the interrupts every clock cycle, which is 20 nanoseconds (ns).

Figure 3-1 GPIO14 Interrupt Scheme



3.2 Register Descriptions

FPGA register descriptions follow.

FPGA Revision Register (0x00)

The **FPGA Revision Register** is an 8-bit read-only register that provides the FPGA major and minor hardware revision information.

The Minor Revision (`MIN_REV`) is indicated on the Diagnostic LEDs immediately at power-up. See the **LED Control Register (0x02)**.

Software clears all bits during a boot-up.

Figure 3-2 FPGA Revision Register @ Base + 00h

Bit 7	6	5	4	3	2	1	Bit 0
MAJ_REV				MIN_REV			

Bit	Field	Description	Value	Access
7-4	MAJ_REV	Major Revision Level – Current major hardware version of the FPGA. This number increments for each major hardware revision.	<major_num>	R/O
3-0	MIN_REV	Minor Revision Level – Current minor hardware revision of the FPGA. This number increments for each minor hardware revision.	<minor_num>	R/O

Board ID and DIP Switch Register (0x01)

The **Board ID and DIP Switch Register** is an 8-bit read-only register that indicates the configuration of the four hard straps on the WANic-66512 and the settings of the S1 DIP Switch. This register also indicates the WANic-66512 boot device.

Figure 3-3 Board ID and DIP Switch Register @ Base + 01h

Bit 7	6	5	4	3	2	1	Bit 0
RSVD		DIP_SW1	DIP_SW0	BOARD_ID 3	BOARD_ID 2	BOARD_ID 1	BOARD_ID 0

Bit	Field	Description	Value	Access
7-6	RSVD	Reserved		
5	DIP_SW1	DIP Switch Indicator 1 – Selects the device from which the WANic-66512 is to boot.	0 = PCIe Bus (On) 1 = Flash (Off)	R/O
4	DIP_SW0	DIP Switch Indicator 0 – Indicates when the Diagnostic Utility is enabled.	0 = Enable (On) 1 = Disable (Off)	R/O
3	BOARD_ID3	Board Identifier – Indicates the heat sink present as either active (fan) or passive.	0 = Active 1 = Passive	R/O
2-0	BOARD_ID[2:0]	Board Identifier – Indicates the hard strap configuration as specified by the build configuration bits.	See Table 3-2	R/O

Table 3-2 Build Configuration Bit Values

Board_ID 2	Board_ID 1	Board_ID 0	Front End Configuration Description
0 (In)	1 (Out)	0 (In)	Two 10G SPF+ with SyncE.

LED Control Register (0x02)

The **LED Control Register** is an 8-bit read/write register that controls the four diagnostic LEDs on the circuit side of the card as shown in Figure 3-3.

The Diagnostic LEDs power up to indicate the FPGA current minor revision (MIN_REV) immediately at power-up (before software loads the board.)

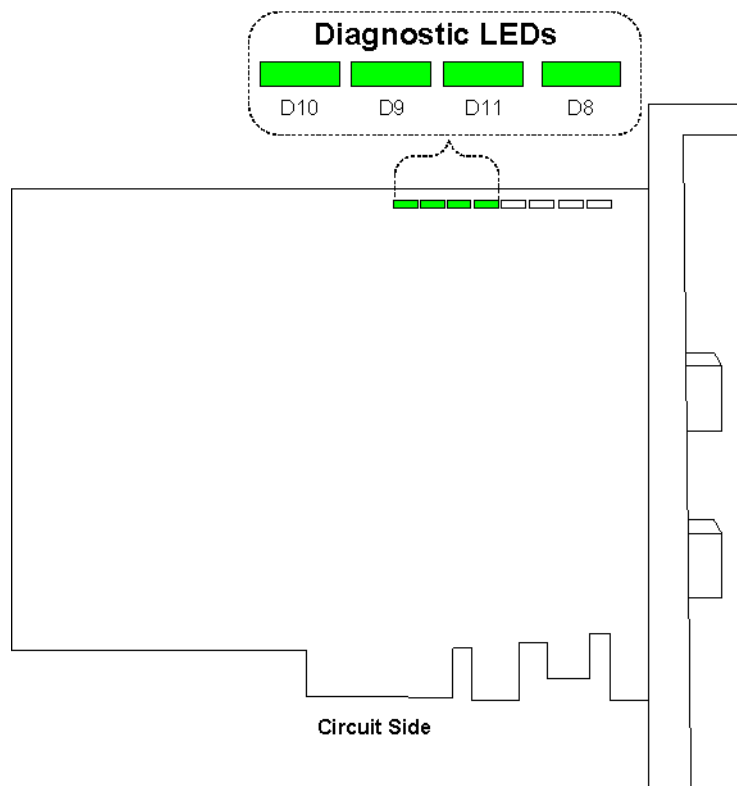
All bits reset to 0 on power-up.

Figure 3-4 LED Control Register @ Base + 02h

Bit 7	6	5	4	3	2	1	Bit 0
RSVD				DIAG LED3	DIAG LED2	DIAG LED1	DIAG LED0

Bit	Field	Description	Value	Access
7-4	RSVD	Reserved		
3-0	DIAG_LED[3:0]	Diagnostic LEDs – Control the diagnostic LEDs by turning these LEDs On or Off.	0 = Off 1 = On	R/W

Table 3-3 Diagnostic LEDs on WANic-66512



Interrupt Control Register (0x03)

The **Interrupt Control Register** is an 8-bit read/write register that generates an interrupt upon completion of a transaction on any of the I2C interface ports. Interrupt control bits function as masks for the corresponding bits in the **Interrupt Status Register**. When a bit in the **Interrupt Status Register** is set, the corresponding bit in the **Interrupt Control Register** is also set. Each register controls the generation of interrupts within its I/O register interface as a function of I2C transactions initiated within that register space. Neither register can control or examine the **Interrupt Control Register** mapped to the other's I/O space.

All bits reset to zero on a hardware reset.

Figure 3-5 Interrupt Control Register @ Base + 03h

Bit 7	6	5	4	3	2	1	Bit 0
FAULT_FAN_IEN	TEMP_IEN	EEPROM_IEN	TEMP_I2C_IEN	RSVD		SFP1_IEN	SFP0_IEN
Bit	Field	Description			Value	Access	
7	FAN_FAULT_IEN	Fan Fault Interrupt Enable – When enabled, generates an interrupt on completion of I2C transactions involving the Fan Controller.			0 = Disable 1 = Enable	R/W	
6	TEMP_IEN	Temperature Interrupt Enable – When enabled, generates a temperature/fan speed interrupt.			0 = Disable 1 = Enable	R/W	
5	EEPROM_IEN	EEPROM Interrupt Enable – When enabled, generates an interrupt to indicate a completed I2C transaction involving the serial EEPROM.			0 = Disable 1 = Enable	R/W	
4	TEMP_I2C_IEN	Temperature Sensor Interrupt Enable – When enabled, generates an interrupt to indicate a completed I2C transaction involving the temperature sensor.			0 = Disable 1 = Enable	R/W	
3–2	RSVD	Reserved					
1–0	SFP[1:0]_IEN	SFP[1:0] Interrupt Enable – When enabled, generates an interrupt to indicate a completed I2C transaction on the specified front panel SFP port.			0 = Disable 1 = Enable	R/W	

Interrupt Status Register (0x04)

The **Interrupt Status Register** is an 8-bit read/write register that indicates the status of a requested I2C transaction for selected devices. An independent **Interrupt Control Register** is available for each I/O bus space. When a bit is set, it indicates that the I2C transaction is completed with or without errors for the specified device.

The host writes a 1 to each bit to clear it. A hardware reset also clears each bit.



NOTE

The **Interrupt Status Register** bits are *always* set (1) when the respective I2C transaction is completed regardless of the value of the bits in the **Interrupt Control Register**.

The **Interrupt Control Register** provides a masking function to determine whether interrupts are generated whenever the corresponding bit is set.

Figure 3-6 Interrupt Status Register @ Base + 04h

Bit 7	6	5	4	3	2	1	Bit 0
FAN_FAULT_INT	TEMP_INT	EEPROM_INT	TEMP_I2C_INT	RSVD		SFP1_INT	SFP0_INT

Bit	Field	Description	Value	Access
7	FAN_FAULT_INT	Fan Fault Interrupt Enable – When set to 1, indicates a MAX6653 fan fault.	0 = Normal 1 = Interrupt	R/W
6	TEMP_INT	Temperature Interrupt Enable – When set to 1, indicates a a MAZ6653 temperature sensor/fan speed fault.	0 = Normal 1 = Interrupt	R/W
5	EEPROM_INT	EEPROM Interrupt Enable – When set to 1, indicates a completed I2C transaction involves the serial EEPROM.	0 = Normal 1 = Interrupt	R/W
4	TEMP_I2C_INT	Temperature Sensor Interrupt – When set, indicates a completed I2C transaction involves the temperature sensor.	0 = Normal 1 = Interrupt	R/W
3–2	RSVD	Reserved		
1–0	SFP[1:0]_INT	SFP[1:0] Interrupt – When set, indicates a completed I2C transaction involves the specified SFP front panel port[1:0].	0 = Normal 1 = Interrupt	R/W

Peripheral Reset Register (0x05)

The **Peripheral Reset Register** is an 8-bit read/write register that allows the Processor to force assertion of the *Reset* signal for selected on-card devices.



NOTE

PHY A maps to Port 0, and PHY B maps to Port 1.

Figure 3-7 Peripheral Reset Register @ Base + 05h

Bit 7	6	5	4	3	2	1	Bit 0
SYNCLK_PWR_B_N	CN66xx_RSTN	RSVD		PHY_B_RSTN	PHY_A_RSTN	DIMMs_RSTN	FLASH_RSTN

Bit	Field	Description	Value	Power Up Setting	Access
7	SYNCLK_PWR_B_N	SyncE Clock Power Reset – When set to 0, issues a hardware reset to the ZL30152 SyncE device. This bit clear to 1 at power-up.	0 = Reset 1 = Normal	1	R/W
6	CN66xx_RSTN	Multi-Core Processor Reset – When set to 0, issues a reset to the Multi-Core Processor This bit clear to 1 at power-up.	0 = Reset 1 = Normal	1	R/W
5-4	RSVD	Reserved			
3	PHY_B_RSTN	PHY B Reset – When set to 0, issues a reset to the PHY B (Port 1) device. This bit clear to '0' at power-up. (For example, the device is left in reset until taken out.)	0 = Reset 1 = Normal	0	R/W
2	PHY_A_RSTN	PHY A Reset – When set to 0, issues a reset to the PHY A (Port 0) device. This bit clear to 0 at power-up. For example, the device is left in reset until taken out.	0 = Reset 1 = Normal	0	R/W
1	DIMMs_RSTN	DIMMs Reset – When set to 0, issues a reset to the registered DIMM modules. This bit clear to 1 at power-up.	0 = Reset 1 = Normal	1	R/W
0	FLASH_RSTN	Flash Reset – When set to 0, issues a reset to the boot Flash. This bit clear to 1 at power-up.	0 = Reset 1 = Normal	1	R/W

Transmit Control Register (0x06)

The **Transmit Control Register** is an 8-bit read/write register that controls the front panel SFP+ transmitter output. A hardware reset sets all bits to zero (disable).

Figure 3-8 Transmit Control Register @ Base + 06h

Bit 7	6	5	4	3	2	1	Bit 0
RSVD						CH1 _EN	CH0 _EN

Bit	Field	Description	Value	Access
7-2	RSVD	Reserved		
1	CH1 _EN	Channel 1 Enable - When set to 1, enables transmitter output for SFP+ 1.	0 = Disable 1 = Enable	R/W
0	CH0 _EN	Channel 0 Enable - When set to 1, enables transmitter output for SFP+ 0.	0 = Disable 1 = Enable	R/W

Fan and Temperature Status Register (0x07)

The **Fan and Temperature Status Register** is an 8-bit read/write register that provides the status for the Temperature Monitor. This register provides the status for the die temperature as well as the temperature of the diode (PN) junction of the Processor.

Figure 3-9 Fan and Temperature Status Register @ Base + 07h

Bit 7	6	5	4	3	2	1	Bit 0
RSVD				SHUTDOWN _REMOTE	SHUTDOWN _LOCAL	FAN_ FAULT	RSVD

Bit	Field	Description	Value	Access
7–4	RSVD	Reserved		
3	SHUTDOWN _LOCAL	Shutdown Local – When set to 0, indicates that the local temperature is <i>above</i> the shutdown setpoint.	0 = Fault 1 = Normal	R/W
2	SHUTDOWN _REMOTE	Shutdown Remote – When set to 0, indicates that the remote temperature is <i>above</i> the shutdown set point.	0 = Fault 1 = Normal	R/W
1	FAN_ FAULT	Fan Fault – When set to 0, indicates that the fan has experienced a fault.	0 = Fault 1 = Normal	R/W
0	RSVD	Reserved		

I2C Registers

The WANic-66512 contains I2C compatible devices including:

- Front panel SFP+ modules
- One EEPROM
- One Temperature Sensor

The FPGA provides a separate I2C port to interface with each device. Each I2C port has a separate set of four I/O registers that include the following:

- **I2C Address High Register**
- **I2C Address Low Register**
- **I2C Data Register**
- **I2C Control Register**

In response to requests from the Processor, the FPGA performs read and write transactions on each I2C port by means of these four registers.

The FPGA permits the Processor to simultaneously initiate transfers to the same device and automatically resolves simultaneous transfer requests. The FPGA carries out simultaneous transfers sequentially. An internal arbitrator performs the following:

- Determines which transfer to process first
- Coordinates the transfer of data between the respective I2C interface and the requesting device
- Notifies the requesting device when the transfer is complete

Each I2C transaction requires a specific sequence of commands for writing and reading data. The value written to the *Least Significant Bit (LSB)* of the **I2C Control Register** determines whether an I2C read or write transaction is requested.

During a transaction, the FPGA reports an I2C error in the **I2C Control Register** when the associated I2C device fails to maintain communications with the I2C interface; for example, when the host tries to access a register at a non-existent address.

Writing Data

To write data in an I2C transaction, perform the following steps:

1. Each I2C transaction requires a register address specification prior to initiating an I2C transaction. The host must specify this address by writing to the **I2C Address Register**. (**I2C High Address** and **I2C Address Low Registers**, when appropriate.)
2. Next, the host must specify the data to be transmitted by writing to the **I2C Data Register**. The **I2C Data Register** holds the data until the transaction completes.
3. After specifying the I2C address (and data), the host initiates an I2C transaction by writing to the **I2C Control Register** with the *Most Significant Bit* (MSB) and *Least Significant Bit* (LSB) set to 1. All other bits must be set to 0 (*81h*).

First, arbitration logic within the FPGA determines when the associated I2C port is idle. Then, the FPGA obtains the address and data and initiates the transaction.

4. To determine if the transaction is complete, the host can do the following:
 - Poll the content of the **I2C Control Register** or **Interrupt Status Register**.
 - Wait for an interrupt, if interrupts are enabled.
5. When the transaction completes, the **I2C Control Register** resets the *I2C_RUN* field (Bit 7) and the *I2C_ERR* field (Bit 6) updates.

Reading Data

To read data from an I2C transaction, perform the following steps:

1. Specify the address of the register that contains the data.
2. Initiate the I2C transaction by writing *80h* to the **I2C Control Register**.
3. Determine if the transaction is complete. The host can do the following:
 - Poll the content of the **I2C Control Register** or **Interrupt Status Register**.
 - Wait for an interrupt, if interrupts are enabled.
4. When the transaction completes, the **I2C Data Register** contains the data that was read from the specified I2C device. The host can now read the data from this register. The **I2C Control Register** resets the *I2C_RUN* field (Bit 7) and the *I2C_ERR* field (Bit 6) updates.

I2C registers descriptions are as follows.

I2C Control Registers (0x08, 0x0C, 0x1C, 0x18)

This register is an 8-bit read/write register that contains I2C transaction status, activity, and error information. Write to the **I2C Control Register** to initiate a read or write transaction for the specified device. Read the **I2C Control Register** to determine the state of a previously initiated transfer.

The **I2C Control Register** address for specific devices is as follows:

<u>Device</u>	<u>Address</u>
Temperature Sensor	08h
EEPROM	0Ch
SFP+ 0	1Ch
SFP+ 1	18h

Figure 3-10 I2C Control Register @ Base + Device Address

Bit 7	6	5	4	3	2	1	Bit 0
I2C_RUN	I2C_ERR	RSVD					I2C_RW

Bit	Field	Description	Value	Access
0	I2C_RW	I2C Transaction Mode – Initiates either an I2C read or write operation for the specified device. For both operations, the I2C_RUN field (Bit 7) must also be set to 'Run' (1) when writing to the I2C Control Register .	0 = Read 1 = Write	R/W
5–1	RSVD	Reserved		
6	I2C_ERR	I2C Error – When set, indicates that the current I2C transaction completed with errors. This bit automatically reset to zero (no errors) on the next I2C transaction for the specified device.	0 = No errors 1 = Errors	R/W
7	I2C_RUN	I2C Initiate Run – When set, initiates an I2C read or write transaction for the specified device.	0 = Inactive 1 = Run	R/W

I2C Address Low Register / I2C Address High Register (0x0A, 0x0E, 0x1E, 0x1A / 0x09, 0x0D, 0x1D, 0x19)

The **I2C Address Low Register** is an 8-bit read/write register that specifies an address of up to 8-bits. For addresses that exceed 8 bits, specify the upper bits in the **I2C Address High Register**.



NOTE

Values written in both registers must be right justified.

The **I2C Address Register** address for specific devices is as follows:

<u>Device</u>	<u>High Address</u>	<u>Low Address</u>
Temperature Sensor	09h	0Ah
EEPROM	0Dh	0Eh
SFP+ 0	1Dh	1Eh
SFP+ 1	19h	1Ah

Figure 3-11 I2C Address Low Register @ Base + Low Address

Bit 7	6	5	4	3	2	1	Bit 0
I2C_A7	I2C_A6	I2C_A5	I2C_A4	I2C_A3	I2C_A2	I2C_A1	I2C_A0

Figure 3-12 I2C Address High Register @ Base + High Address

Bit 7	6	5	4	3	2	1	Bit 0
I2C_A15	I2C_A14	I2C_A13	I2C_A12	I2C_A11	I2C_A10	I2C_A9	I2C_A8

where **I2C_A[15:0]** is the device address.

I2C Data Registers (0x0B, 0x0F, 0x1F, 0x1B)

The **I2C Data Register** contains the data associated with each I2C transaction. This register is an 8-bit read/write register. Figure 3-13 provides the format for this register.

- For write operations, specify the data by writing to this register *prior* to initiating the transaction.
- For read operations, this register contains data read from an I2C device when the transaction is completed.

The **I2C Data Register** address for specific devices is as follows:

<u>Device</u>	<u>Address</u>
Temperature Sensor	0Bh
EEPROM	0Fh
SFP+ 0	1Fh
SFP+ 1	1Bh

Figure 3-13 I2C Data Register @ Base + Device Address

Bit 7	6	5	4	3	2	1	Bit 0
I2C_D7	I2C_D6	I2C_D5	I2C_D4	I2C_D3	I2C_D2	I2C_D1	I2C_D0

where **I2C_D[7:0]** is data.

Boot Flash Upper Address Control Register (0x20)

The **Boot Flash Upper Address Control Register** is an 8-bit read/write register that provides the upper address location for the boot Flash to break the 1GB Flash into 16 MB pages.

Figure 3-14 Boot Flash Upper Address Control Register @ Base + 20h

Bit 7	6	5	4	3	2	1	Bit 0
RSVD					A26	A25	A24

Bit	Field	Description	Value	Access
7-3	RSVD	Reserved		
0-2	A26-A24	Page number	000-111	R/W

Miscellaneous Functions Register (0x21)

The **Miscellaneous Functions Register** is an 8-bit read-only register that reports the following miscellaneous functions:

- Assigns lane reversal to QLM0.
- Controls the diagnostic clock from the Processor.

At reset, each bit is set to zero.

Figure 3-15 Miscellaneous Functions Register @ Base + 21h

Bit 7	6	5	4	3	2	1	Bit 0
RSVD				QLM0 _REV_LANES	RSVD		MSC_ CLKOUT_EN

Bit	Field	Description	Value	Access
7-4	RSVD	Reserved		
3	QLM0 _REV_LANES	QLM0 Lane Reversal - When enabled, assigns lane reversal for QLM0.	0 = Disable 1 = Enable	R/W
2-1	RSVD	Reserved		
0	MSC_ CLKOUT_EN	Clock Output Enable - When enabled, assigns the diagnostic clock out of the Processor running at core clock speed.	0 = Disable 1 = Enable	R/W

PHY Interrupt Register (0x22)

The **PHY Interrupt Register** is an 8-bit register that allows the PHYs to generate an interrupt to the Processor as described in [Section 3.1.1 "Temperature Sensor Interrupts"](#).



NOTE

PHY_A map to physical Port 0 and PHY_B maps to Port 1.

Figure 3-16 PHY Interrupt Register @ Base + 22h

Bit 7	6	5	4	3	2	1	Bit 0
PHY_A_MSK	PHY_B_MSK	RSVD		NOT_PHY_A_INTN	NOT_PHY_B_INTN	RSVD	

Bit	Field	Description	Value	Reset	Access
1-0	RSVD	Reserved			
2	NOT_PHY_B_INTN	PHY B Interrupt Inverted – When a '1' is registered in the FPGA, this bit indicates an active interrupt from PHY B.	0 = De-asserted 1 = Asserted	0	R/O
3	NOT_PHY_A_INTN	PHY A Interrupt Inverted – When a '1' is registered in the FPGA, this bit indicates an active interrupt from PHY A.	0 = De-asserted 1 = Asserted	0	R/O
5-4	RSVD	Reserved			
6	PHY_B_MSK	PHY B Masked – When asserted, indicates that PHY B is masked.	0 = De-asserted 1 = Asserted	0	R/W
7	PHY_A_MSK	PHY A Masked – When asserted indicates that PHY A is masked.	0 = De-asserted 1 = Asserted	0	R/W

Thermal Interrupt Mask Register (0x23)

The **Thermal Interrupt Mask Register** is an 8-bit read/write register that allows the Temperature Sensor to generate interrupts to **GPIO14** as described in [Section 3.1.1 "Temperature Sensor Interrupts"](#).

Figure 3-17 Thermal Interrupt Register @ Base + 23h

Bit 7	6	5	4	3	2	1	Bit 0
RSVD	THERM_A_MSK	THERM_B_MSK	THERM_C_MSK	RSVD	NOT (THERM_A_INTN)	NOT (THERM_B_INTN)	NOT (THERM_C_INTN)

Bit	Field	Description	Value	Access
7	RSVD	Reserved		
6	THERM_A_MSK	Therm A Mask — The active high mask for THERM_A_INTN.	0 = Not masked 1 = Masked	R/W
5	THERM_B_MSK	Therm B Mask — The active high mask for THERM_B_INTN.	0 = Not masked 1 = Masked	R/W
4	THERM_C_MSK	Therm C Mask — The active high mask for THERM_C_INTN.	0 = Not masked 1 = Masked	R/W
3	RSVD	Reserved		R/O
2	NOT (THERM_A_INTN)	Not Therm A Interrupt — The active low interrupt is inverted and registered here for FAN_FAULT from the Temperature Sensor. This indicates a fan fault.	0 = Normal 1 = Fan Fault	R/O
1	NOT (THERM_B_INTN)	Not Therm B Interrupt — The active low interrupt is inverted and registered here for FAN_THERM from the Temperature Sensor. This indicates that the Over Temperature setpoint has been exceeded.	0 = Normal 1 = Over Temp.	R/O
0	NOT (THERM_C_INTN)	Not Therm C Interrupt — The active low interrupt is inverted and registered here for PHY C.	0 = Normal 1 = Alert	R/O

IR3541 Interrupt Register (0x24)

The **IR3541 Interrupt Register** is an 8-bit read-only register that indicates when the IR3541 Multiphase Controller detects an alarm condition (*alert*) or an over-temperature (*hot*) condition for the DDR3 memory or the Processor.

Figure 3-18 IR3541 Interrupt Register @ Base + 24h

Bit 7	6	5	4	3	2	1	Bit 0
RSVD				NOT (MEM_ALERT)	NOT (MEM_HOT)	NOT (CORE_ALERT)	NOT (CORE_HOT)

Bit	Field	Description	Value	Access
7–4	RSVD	Reserved		R/O
3	NOT (MEM_ALERT)	Memory Alerts — When set to '1', indicates a new V12 status (typically, VR settled, ICC Max, or temperature alert.)	0 = Normal 1 = Alert	R/O
2	NOT (MEM_HOT)	Memory Hot — When set to '1', indicates that the maximum temperature is exceeded,	0 = Normal 1 = Over temperature	R/O
1	NOT (CORE_ALERT)	Core Alerts — When set to '1', indicates a new V12 status (typically, VR settled, ICC Max, or temperature alert.)	0 = Normal 1 = Alert	R/O
0	NOT (CORE_HOT)	Core Hot — When set to '1', indicates that the maximum temperature is exceeded.	0 = Normal 1 = Over temperature	R/O

DIMM Thermal Event Register (0x25)

The **DIMM Thermal Event Register** is an 8-bit register that stores the thermal event signal from each DIMM with an aligned mask bit as described in [Section 3.1.1 "Temperature Sensor Interrupts"](#).

Figure 3-19 DIMM Thermal Register @ Base + 25h

Bit 7	6	5	4	3	2	1	Bit 0
RSVD		DIMM_SLOT2_ MSK	DIMM_SLOT1_ MSK	RSVD		NOT (DIMM_SLOT2_ TEVENT_INTN)	NOT (DIMM_SLOT1_ TEVENT_INTN)

Bit	Field	Description	Value	Access
7–6	RSVD	Reserved		
5	DIMM_SLOT2_ MSK	DIMM Slot 2 Interrupt – When enabled, masks the interrupt on DIMM Slot 2.	0 = Disable 1 = Enable	R/W
4	DIMM_SLOT1_ MSK	DIMM Slot 1 Interrupt – When enabled, masks the interrupt indicates on DIMM Slot 1.	0 = Disable 1 = Enable	R/W
3–2	RSVD	Reserved		
1	NOT (DIMM_SLOT2_ TEVENT_INTN)	DIMM Slot 2 Thermal Event Interrupt – The interrupt is inverted and registered here to indicate a fault on DIMM Slot 2.	0 = No fault 1 = Fault	R/O
0	NOT (DIMM_SLOT1_ TEVENT_INTN)	DIMM Slot 1 Thermal Event Interrupt – The interrupt is inverted and registered here to indicate a fault on DIMM Slot 1.	0 = No fault 1 = Fault	R/O

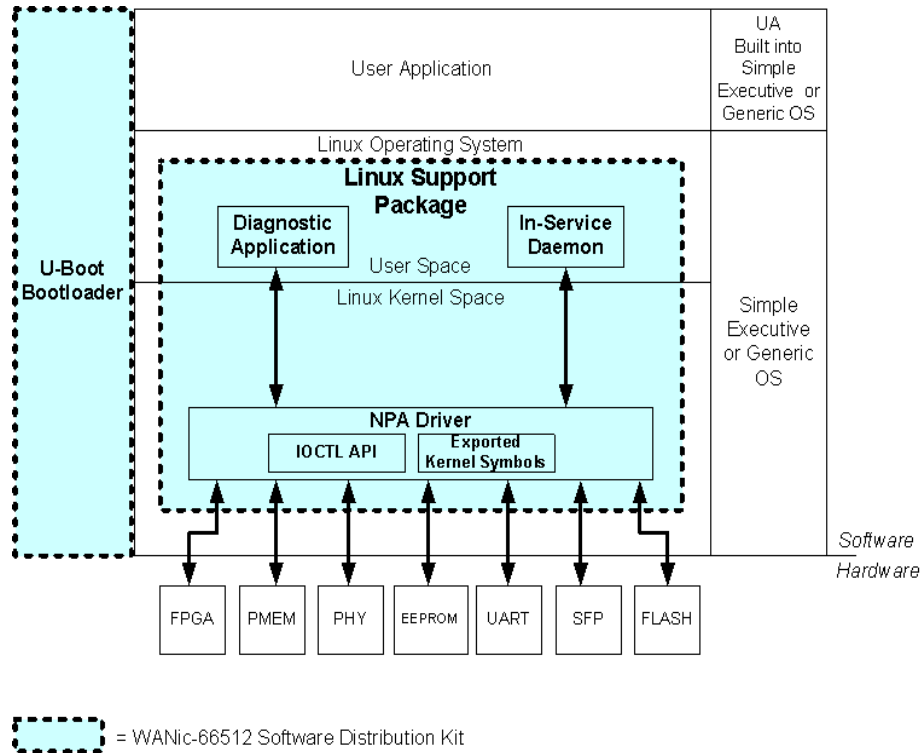
4 • Software Description

The Processor (and the Simple Executive) on the WANic-66512 support embedded Linux-based boot firmware and application software. The software manages the hardware, configuration and monitoring of data processing, and also provides services to the host. The software executes on the WANic-66512 and is accessible through an IOCTL Application Programming Interface (API). The API provides easy software integration for creating customized applications to configure and to monitor the WANic-66512. The software makes it easy to configure a variety of signaling, gateway, real-time control, traffic processing, and network interface applications.

4.1 Software Overview

The WANic-66512 software executes on one or more cnMIPs64 cores, which are specified by the user. The WANic-66512 Software Development Kit consists of the components shown in Figure 4-1.

Figure 4-1 Software Components



- User Application — controls and manages processing for the WANic-66512.
- U-Boot Bootloader – based on the Universal Bootloader (U-Boot) provides initialization and reset operations, and operating system and application load.
- Linux Support Package (LSP) – provides a suite of functions and drivers to access and to use the WANic-66512 hardware and provides a platform for the Control Plane.
- Linux Operating System – Debian™ distribution with Cavium OCTEON support.
- Wind River PNE-LE Board Support Package (BSP) — provides a suite of functions to access and to use the WANic-66512 hardware.
- Diagnostic and Configuration Utility – Linux image that provides tests to configure and to verify the WANic-66512 hardware. (See “[Chapter 5: Linux Support Package \(LSP\) Applications](#)” for more information.)

The WANic-66512 distribution software also contains the following GNU components:

- GNU Tool-Chain – provides tools for building executables for the Processor.
- GNU Debugger – is a standard debugger for the GNU software that helps diagnose a program as it is executing.



NOTE

See “[Appendix B: GNU General Public License V2](#)” for software licensing information.

In addition, obtain one of the following operating systems from an external source:

- Wind River PNE-LE Linux, Version 4.3
- Debian cnMIPs Linux
- OCTEON Simple Executive



NOTE

See the Cavium Network User’ site at www.cnusers.org for more information about the OCTEON Simple Executive.

4.2 Cavium Software Development Kit

The Cavium Software Development Kit (SDK) is included on the WANic-66512 distribution CD-ROM for rapid development of the Processor. The SDK includes drivers, libraries, files, and other tools to facilitate application development. The SDK is covered by the GNU General Public License version 2. (See “[Appendix B: GNU General Public License V2](#)” for more information.) This section briefly describes selected SDK features.

4.2.1 OCTEON Ethernet Driver

The Cavium SDK provides the OCTEON Ethernet Driver (`octeon-ethernet`), which is found in the following directory:

```
$OCTEON_ROOT/linux/kernel_2.6/linux/drivers/net/octeon-ethernet
```

The default Cavium Linux kernel configuration builds the OCTEON Ethernet Driver as a module. For instructions on building the embedded Linux kernel, see the WANic-66512 software CD-ROM.

After the OCTEON Ethernet Driver is loaded on the Processor, two new Ethernet interfaces are available: `Xau0` and `Xau1`. These two ports correspond to the front panel ports in the following order:

- `Xau0` -> Port 0
- `Xau1` -> Port 1

Both Ethernet ports function as standard Linux Ethernet interfaces, and can be configured with Linux tools, such as `ifconfig` and `ethtool`¹.

The Ethernet PCI interfaces (`pci0`, `pci1`, `pci2` and `pci3`) are also available when EtherPCI is enabled in the software installation process.



NOTE

For instructions on building and configuring Ethernet over PCI, see the software installation instructions on the WANic-66512 software CD-ROM. This CD-ROM contains all the components to support software development for the OCTEON Multi-Core Processor, or the Embedded Debian Linux environment on a WANic-66512 packet processor.

1. `Ethtool` and `ifconfig` are utilities that allows querying and changing of Ethernet card settings, such as speed, port, auto-negotiation, and PCI locations.

4.2.2 Cavium Embedded File System

The Cavium SDK provides the Embedded File System, which is contained in the following directory:

```
$OCTEON_ROOT/linux/embedded_rootfs
```

The Embedded File System is a RAM-based file system, which uses BusyBox (an open-source software application licensed under the *GNU General Public License*, version 2) as its base, and also supplies the optional Linux applications and utilities listed in Table 4-1.

Table 4-1 Linux Applications and Utilities

Utility	Description
libpcap	System-independent interface for user-level packet capture. libpcap provides a portable framework for low-level network monitoring. Applications include network statistics collection, security monitoring, network debugging, and so forth.
libpopt	Library that parses command-line options.
bridge-utils	Bridge building package that configures a Linux Ethernet bridge, which connects two or more physical Ethernet devices.
ethtool	Linux command for displaying or modifying network drivers and hardware.
flex	Tool for generating scanners. A scanner, sometimes called a tokenizer, is a program which recognizes lexical patterns in text. The flex program reads user-specified input files, or its standard input if no file names are given, for a description of a scanner to generate.
ipsec-tools	Internet protocol security (IPsec) that includes utilities to manipulate IPsec connections with the Linux-2.6.x.
iptables	Generic table structure that defines rules and commands as part of the netfilter framework that facilitates Network Address Translation (NAT) packet filtering, and packet mangling in the Linux 2.4 and later operating systems.
iproute2	Collection of utilities for controlling TCP/IP networking and traffic control in Linux.
mii-tools	Checks or sets the status of a network interface's Media Independent Interface (MII) Ethernet device.
mtd-tools or should this be mtd-utils	Utilities for manipulating memory devices, such as Flash memory, Disk-On-Chip, or ROM. Includes mkfs.jffs2, a tool to create JFFS2 (Journaling Flash File System) file systems.
net-tools	Comprehensive set of Linux networking tools that provide host monitoring, network scanning, security, administration tools and much more with a user interface.
openSSL	Implementation of the SSL and TLS protocols.
pciutils	Contains various Linux utilities for inspecting and setting of devices connected to the PCI bus.
readline	Line-editing and history capabilities for interactive programs with a command-line interface. It allows users to move the text cursor, search the command history, control a kill ring (a more flexible version of a copy/paste clipboard) and use tab completion on a text terminal.

Table 4-1 Linux Applications and Utilities

Utility	Description
strace	Debugging utility for Linux to monitor the system calls used by a program and all the signals it receives.
tcpdump	Command-line tool for monitoring (sniffing) network traffic. Tcpdump works by capturing and displaying packet headers and matching them against a set of criteria.
wireless-tools	Package contains the Wireless tools, used to manipulate the Linux Wireless Extensions. The Wireless Extension is an interface allowing you to set Wireless LAN specific parameters and get the specific stats.
zlib	Compression library provides in-memory compression and decompression functions, including integrity checks of the uncompressed data.
OCTEON	Package of utilities to support the OCTEON cnMIPs cores.
Toolchain	Collection of programming tools produced by the GNU Project. These tools form a toolchain (suite of tools used in a serial manner) used for developing applications and the operating systems.



NOTE

For more information on BusyBox, see <http://www.busybox.net>.

To configure the Embedded Linux File System, run the following commands:

```
# cd $OCTEON_ROOT/linux/embedded_rootfs
# make menuconfig
```

Configure the file system using the WANic-66512 configuration menu, Kconfig, which is similar to the traditional Linux Kconfig Menu. Once the configuration is completed, Save the changes and Exit.

For instructions on building the newly configured Embedded File System, see the installation instruction on the WANic-66512 software CD-ROM.

4.2.3 Simple Executive Library

The Cavium SDK provides the Simple Executive (Exec) Library, which is contained in the following directory:

```
$OCTEON_ROOT/executive
```

The Simple Exec Library is a runtime library that also provides a hardware abstraction layer across all OCTEON Processors. For an overview of the Simple Exec Library, see the file:

```
$OCTEON_ROOT/executive/README.txt
```

Simple Exec sample applications are provided and contained in the directory:

```
$OCTEON_ROOT/cav-gefes/examples/simple_exec
```

For instructions on how to build the examples, see [Section 4.7 Examples](#) in this chapter.

4.3 User Application

The User Application (UA) is the software component that controls and manages processing. The WANic-66512 U-Boot loads and boots the UA. The UA operation is distributed across the multiple cores of the Processor.

The UA can be composed of:

- A unique set of tasks performed by each core, or
- Similar or identical tasks distributed across multiple cores in an attempt to balance core loading

The UA should always strive to prevent core spin-locks and idle time during high loading periods among the cores.

The UA can interface to the LSP when it configures or monitors the WANic-66512 hardware.

The UA can also interface to the LSP when it runs foreground or background diagnostics for WANic-66512 hardware testing. The UA can assert the LSP API for configuring, monitoring, programming, and testing operations of the WANic-66512 hardware.

4.4 U-Boot BootLoader

The WANic-66512 U-Boot Bootloader is based on the U-Boot open source multi-platform bootloader, which is used for a wide range of embedded processor architectures. The WANic-66512 U-Boot bootloader (hereafter referred to as U-Boot) supports interactive commands, environment variables, command, and Flash located user application read/write.

U-Boot can be started either from local Flash memory or remotely over the PCI Express (PCIe) bus. U-Boot transfers itself to local RAM where it completes execution. Then, U-Boot loads the UA either from a remote TFTP server or from local Flash memory, or remotely over the PCIe bus. After loading the UA into RAM, U-Boot boots the operating system on the specified MIPS64 core(s). U-Boot executes only on MIPS64 Core 0.

4.4.1 Boot Process

The WANic-66512 contains the following U-Boot image on Flash:

- Normal image – Standard WANic-66512 Flash bootloader.

During the boot process, U-Boot executes from Flash on Core 0 only. A system reset transfers execution to the Normal image immediately. After reset, U-Boot transfers execution to RAM. In RAM, it conducts the operations dictated by the contents of EEPROM. U-Boot supports non-volatile configuration retrieval.

U-Boot software performs basic initialization of the WANic-66512. When this initialization is complete, U-Boot checks the application images configuration settings in Non-Volatile RAM (NVRAM) to determine which application images to load. The application images are decompressed from the network or Flash, and loaded into memory. When an application image is not found, U-Boot enters Command Line Mode and waits for user input.

Once the images are validated and loaded to memory, U-Boot transfers control to the UA software and also passes environment variables that are relevant to the application images. (See “[Section 4.4.7 U-Boot Scripting](#)” for information on environment variables.) U-Boot provides special commands for displaying information and performing various operations. U-Boot commands are described in Table 4-3.

Tuples U-Boot uses *tuples* to define and to store the configuration of the WANic-66512 in EEPROM. (A *tuple* is a value stored in EEPROM that defines board or test settings, and stored error codes for various tests.) The WANic-66512 U-Boot extends the tuple configuration definitions to U-Boot commands and includes new tuples for the WANic-66512 as described in the section “[4.4.11 EEPROM Tuple Update and Enumeration](#).”

POST U-Boot also performs Power On Self Test (POST) operations after initialization. POST operations validate the hardware integrity.



NOTE

POST operations run only when Switch 4 on the DIP Switch S1 bank is On, or if the 'postall' environment variable is set. See "[Section 2.11 Dual In-Line Package \(DIP\) Switch](#)" for more information on the S1 DIP Switch bank.

The WANic-66512 provides the following short and long POST tests:

- Flash checksum – validates a Flash checksum on an individual sector or a collection of sectors
- RAM – validates RAM integrity
- PHY – validates PHY integrity
- PMEM – validates PMEM integrity
- DIMM0 – validates DIMM0 interface access via SPD
- DIMM1 – validates DIMM1 interface access via SPD

POST failure results in error code indications. The resulting action is determined by the associated test failure for that particular test. Failure action can include the following:

- System reset
- System halt
- System partial continuation to the U-Boot menu for manual debugging

After a power loss, POST results are stored in 16 bytes of EEPROM (as shown in Figure 4-3) to indicate the failures described in Table 4-2.

Table 4-2 POST Failures

Value	Description
FF	No error
00	Flash Checksum Failure
01	RAM Failure
02	PHY Failure
03	SerDes Failure
04	RLDRAM Failure
05	DIMM0 Failure – Not present
06	DIMM1 Failure – Not present
08	PMEM Failure
80	Diagnostic Failure

4.4.2 Building U-Boot

U-Boot can be configured for Flash-based execution, or for execution through the PCI Express interface.

Flash Boot

To build U-Boot for Flash boot, enter the following commands at the build system prompt in the U-Boot source directory:



NOTE

This booting method supports only a single executable image, which can be booted on Core 0 only.

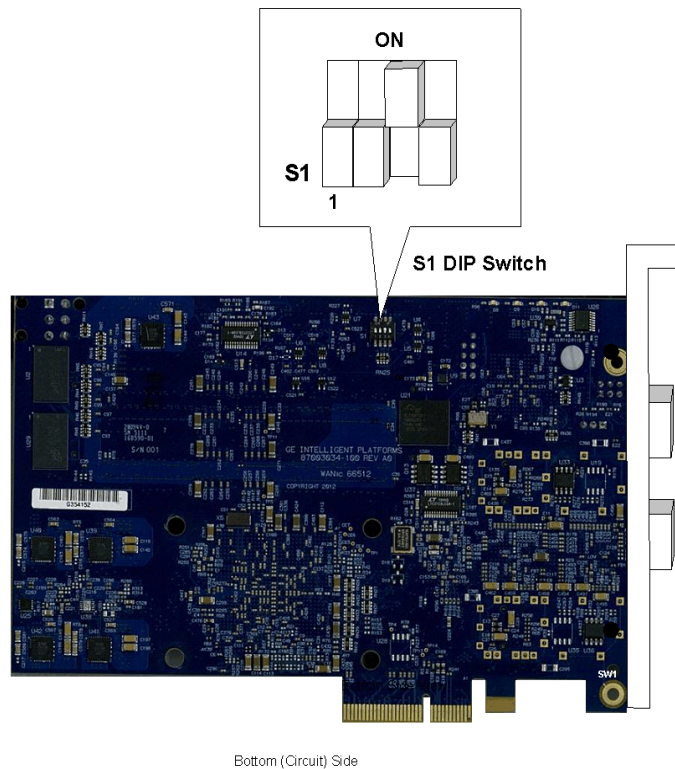
1. Configure U-Boot to build for Flash-based execution by entering the following command:
make octeon_w66xx_config
2. Build the Flash-based U-Boot image using the make command.
make
3. Using the Flash Menu in the Diagnostic Utility, copy the Flash-based boot image to the on-board Flash.

Re-Programming U-Boot

Re-program U-Boot using the S1 bank of DIP Switches as shown in Figure 4-2. To re-program U-Boot, perform the following steps:

1. Set Switch 4 to the 'On' position.
2. Run the npaDiag application in the embedded Linux pre-loaded in the Flash.

Figure 4-2 S1 DIP Switch Location



4.4.3 U-Boot Commands

Table 4-3 lists U-Boot commands in alphabetical order. For more information on these commands, see the *DENX U-Boot and Linux Guide (DULG)* <http://www.denx.de/wiki/DULG/Manual>.

Table 4-3 U-Boot Commands

Command	Description
?	Displays online help.
askenv	Gets the environment variables from "stdin".
autoscr	Runs the Shell script under U-Boot from memory.
base	Displays or sets a base address that is used as an address offset for memory commands. The default value of the base address is 0.
bootelf	Boots from an ELF image in memory.
bootoct	Boots from the OCTEON Executive Elf image in memory.
bootoctelf	Boots a generic Elf image in memory. Note: This command does not support Simple Executive applications. Instead use the <code>bootoct</code> command.
bootoctlinux	Boots from a Linux ELF image in memory.
bootp	Boots the image over the network using the BOOTP/TFTP protocol.
cmp	Compares the contents of two memory areas.
coninfo	Displays information about the available console I/O devices. The output contains information such as device name, flags, current usage, and so forth.
cp	Copies memory areas.
crc32	Calculates a CRC-32 checksum over a range of memory.
dhcp	Invokes the DHCP client to obtain the IP/boot parameters.
defaultenv	Uses default environment variables.
echo	Displays the same argument typed on the console.
eeeprom	EEPROM sub-system.
erase	Erases the contents of one or more seconds of the Flash memory.
fatinfo	Prints information about the file system.
fatload	Loads binary file from a DOS file system.
fatloadalloca	Loads binary files from a DOS file system and allocates a named bootmem block for file data.
fatls	Lists files in a directory.
flinfo	Displays information for all available Flash memory banks.
fr	Reads the Flash.
freeprint	Prints a list of free bootmem blocks.
go	Starts standalone applications at address "addr."
gunzip	Un-compresses an in-memory gzipped file.
help	Displays online help. Without arguments, it displays a short listing of all available U-Boot commands.
ide	IDE sub-system.
itest	Returns true or false on an integer compare.
loadb	Loads a binary file over serial lines (kermit mode).

Table 4-3 (continued) U-Boot Commands

Command	Description
loop	Performs an infinite loop on an address range. To stop the loop, reset the card.
md	Displays memory contents both as hexadecimal and ASCII data.
mii	MII Utility command.
mm	Interactively modifies the memory contents.
mtest	Performs a simple Random Access Memory test. This test fails when applied to ROM or Flash.
mw	Initializes (fills) memory with some value.
namealloc	Allocates a named bootmem block.
namedfree	Frees a named bootmem block.
namedprint	Prints a list of named Boolean blocks.
netintstat	Obtains network interface status.
nm	Memory modify (constant address).
pcie <flag> <addr>	Accesses the PCI Express, where: <flag> = r(read), w(rite), or d(lump). <addr> = address For example, the command, <code>pcie 0x001</code> , reads from PCI Express Configuration Register at address 0x001.
phy <flag> <phyaddr> <phyreg><value>	Accesses the PHY, where: <flag> = r(read), w(rite), or d(lump) <phyaddr> = Address of PHY <phyreg> = PHY register <value> = Write-only value For example, the command <code>phy w 0x40 17 0x1234</code> , writes to Phy Register 17 at address 0x40.
phyl <n> <location>	Toggles the Line or Mac loopback for internal Phy testing or toggles Transmit Enable for Phy testing, where: <n> = PHY number <location> = l(line) or m(ac) or t(ransmit enable)
ping	Sends ICMP ECHO_REQUEST to network host.
printenv	Displays one, several or all variables of the U-Boot environment.
protect	Enables or disables Flash protection. Sets certain parts of the Flash memory to read-only mode, or makes the Flash memory writable again.
rarpboot	Boots the image over the network using RARP/TFTP protocol.
read64	Reads a 64-bit word from 64-bit address.
read64b	Reads a 8-bit word from 64-bit address.
read64l	Reads a 32-bit word from 64-bit address
read64w	Reads a 16-bit word from 64-bit address
read_cmp	Reads and compares memory to value.
readi2c	Reads data from an I2C device.
readpci	Reads data from a PCI device.
readpmem	Reads data from Persistent Memory.
reset	Reboots the system.
run	Runs commands in an environment variable.
saveenv	Saves the environment variables to persistent storage.
sdump <reg>	Dumps status registers, where: <reg> = g(MX) or p(CX)

Table 4-3 (continued) U-Boot Commands

Command	Description
setenv	Sets the environment variables.
sfinit	Initializes the SFPs.
sfpslow	Reads and displays current SFP installation.
sleep	Delays execution for a specified number of seconds (in decimal).
testmem	Tests memory.
testpmem	Tests Persistent Memory.
tftpboot	Boots the image over the network using the TFTP protocol.
tlv_eeeprom	EEPROM data parsing for the module.
validatenfi	Validates the Normal Flash image.
version	Displays the monitor version and build date of the U-Boot image running on your system.
write64	Writes 64-bit words to a 64-bit address.
write64b	Writes 8-bit words to a 64-bit address.
writei2c	Writes data to an I2C device.
write64l	Writes 32-bit word to a 64-bit address.
write64w	Writes 16-bit word to a 64-bit address.
writepci	Writes data to a PCI device.
writepmem	Writes data to Persistent Memory.

4.4.4 PCI Console Redirection – U-Boot

PCI console redirection allows all serial output to be redirected to PCI host, where users can run PCI host tool to take serial input and to display serial output.



NOTE

Host utils in Cavium SDK 2.0 are located in `host/remote-utils/pct-remote-*`.

To set U-Boot for PCI Console Redirection, perform the following steps from the Linux prompt on the Host platform:

1. Once U-Boot has loaded, set environment variable `pci_console_active` to `yes`:

```
# setenv pci_console_active yes  
# saveenv
```
2. Reset the bootloader. U-Boot loads, and informs you that you are:
`'Using PCI console, serial port disabled.'`
3. Before using any host remote tools, make sure that on the PCI host the `OCTEON_REMOTE_PROTOCOL` is set for the specified PCI target device

```
# export OCTEON_REMOTE_PROTOCOL=PCI:0
```
4. Run the remote PCI console application to access U-Boot serial output:

```
# host/remote-tools/oct-remote-console --noraw 0
```

Output is now redirected to the PCI host.

Use `<CTRL+C>` to break out from the PCI host (only in `--noraw` mode).

4.4.5 PCI Console Redirection – Linux

PCI console redirection allows all serial output to be redirected to PCI host where user can run PCI host tool to take serial input and to display serial output

To set Linux for PCI Console Redirection, perform the following steps:

1. Once U-boot has loaded, set environment variable `pci_console_active` to 'yes'.

```
# setenv pci_console_active yes
# saveenv
```

2. Reset the bootloader. U-Boot loads, and provides output to inform you: 'Using PCI console, serial port disabled.'

3. Before using any host remote tools, make sure that on the PCI host the `OCTEON_REMOTE_PROTOCOL` is set for the specified PCI target device

```
# export OCTEON_REMOTE_PROTOCOL=PCI:0
```

4. When booting the Linux kernel image, pass in the kernel parameter `console=pci` to have all output redirected to the PCI host:

```
# bootoctlinux 0 coremask=3ff console=pci
```

5. Run the remote pci console application to access U-Boot serial output:

```
# host/remote-tools/oct-remote-console --noraw 0
```

Output is now redirected to PCI host.

Use <CTRL+C> to exit from the PCI host (only in --noraw mode).

4.4.6 Environment Variables

Once the software image is validated and loaded to memory, U-Boot transfers control to the application software and also passes environment variables that are relevant to the application image. Environment variables control the Flash-based application boot process. Environment variables can be modified from default values listed in Table 4-4, where 'h' represents a hexadecimal default value.

Table 4-4 Environment Variables

Variable	Default	Description
autoload	Yes	Auto loads DHCP and TFTP.
baudrate	115200	Defines the serial console baud rate.
bootloader_flash_update	protect off 0xb7e00000 0xb7efffff;erase 0xb7e00000 0xb7effrfff;cp.b \$(fileaddr) 0xb7e00000 0xf0000 validatenfi	Programs the Normal U-Boot image into Flash following TFTP transfer into memory. Use with the 'run' command.
ddr_pmem	No	Control mechanism for choosing to use physical PMEM, or allocating a section of DDR3 memory as PMEM. Setting this to 'yes' uses a section of DDR3 memory as PMEM instead of the physical PMEM on the board.
ethact	octeth0	Sets the current active Ethernet interface.
mdio_alloc	Yes	Creates shared named alloc for mdio global lock required when using both npaDriver and Cavium-Ethernet driver together.
pci_console_active	<i>Not set by default</i>	Enables PCI console redirection. Use with oct-remote-console.
swfb_active	None	Activates the Flash application boot process.
swfb_cmd	autoscr	Starts the application.
swfb_cmd_arg		Indicates a command argument list passed to the application.
swfb_flash_addr	0x18000000h	Identifies the Flash address of the application.
swfb_image_size	2600000h	Indicates the application image size.
swfb_ram_addr	0x2000000h	Provides the starting RAM address for the application to be copied and executed. When this value is not specified, no copy operation is performed and the image is executed from the Flash address.



NOTE

If during the attempt to obtain Environment Variables, U-Boot encounters an invalid CRC, the environment variables are not retrieved from Flash and a default environment is used. A message similar to the following displays:

```
*** Warning - bad CRC, using default environment
```

To save the environment variables, perform the following steps:

1. Modify the environment.
2. Issue the U-Boot **saveenv** command to store the changes into non-volatile memory. The **saveenv** command burns the default RAM-based environment variables into Flash. The **saveenv** command also burns a CRC into Flash, which is used to test the integrity of the Flash-based IP Configuration Acquisition with DHCP.

4.4.7 U-Boot Scripting

U-Boot provides an alternative solution for performing predefined operations during initialization. U-Boot allows the storage of commands or command sequences in an EEPROM scripts that executes during initialization. U-Boot provides storage for up to four scripts; however, only one script can be active at any given time.

Although scripting performs predefined operations during initialization, it does not provide error recovery procedures. Since error recovery can be critical to the successful load and boot of an application, the SIPI/PFI/SFI/LABI based application provides an alternative to scripting. To create and to update a script tuple in EEPROM, see “[Section 4.4.11 EEPROM Tuple Update and Enumeration](#)” in this chapter.

SIPI/PFI/SFI/LABI Application

U-Boot permits unique Source IP Information (SIPI) for each Ethernet interface. Additionally, U-Boot can apply the same source IP address information to all Ethernet interfaces. U-Boot primarily applies Primary File Information (PFI) and Secondary File Information (SFI), if necessary, to acquire the UA once the source IP address is determined. Then, U-Boot applies Load And Boot Information (LABI) to load and boot the application on user-specified cores.

See the subsection in this chapter entitled “[EEPROM Mapping and Tuples](#)” for information to create SIPI, PFI, SFI, and LABI tuples.

4.4.8 DHCP

U-Boot uses the networks Dynamic Host Control Protocol (DHCP) for IP configuration acquisition and for booting.

IP Configuration Acquisition

DHCP automatically adds an Ethernet device to your network by detecting and assigning an IP address to each connected device. However, some networks do not incorporate DHCP. These networks are referred to as ‘static’ and require you to assign an IP address to each device manually.

U-Boot uses the DHCP protocol on a designated Ethernet port. U-Boot ceases DHCP assertion when the IP configuration is obtained. IP configuration includes the following:

- IP address
- Subnet mask
- Default gateway
- Default TFTP server

The following configuration controls U-Boot IP configuration acquisition:

- DHCP Enable (*Default* = Enable)
- DHCP Ethernet Port ID (*Default* = Port0 GbE SerDes)

U-Boot supports an asynchronous serial port for configuration and status. Therefore, it permits static implementation of the IP configuration and permits DHCP to be disabled.

Command Scripts

A U-Boot command script can be transferred from the DHCP server, loaded into RAM via a TFTP transfer, and executed. The U-Boot *command script* is a collection of U-Boot commands that typically load one or more core images into RAM and then boots them.



NOTE

The U-Boot script is constructed using the mkimage tool on a Linux host. Put the output file in the TFTP directory of the DHCP server.

The following commands configure for simple script loading:

```
setenv bootfile ubootscript.img
setenv loadaddr 2000000
setenv autoload yes
setenv autoscript yes
saveenv
```

The following example is a typical U-Boot command script:

```
setenv serverip 192.168.201.10
setenv nfsaddr 192.168.201.74
setenv mountpnt cava-dev
tftp 20000000 vmlinux.dev
bootoctlinux 20000000 coremask=ffff root=/dev/nfs
ip=$(ipaddr):$(nfsaddr):$(nfsaddr):255.255.255.0:(mountpnt)
:eth0 panic=2
```

Encoded DHCP Server Response

DHCP provides a mechanism for obtaining the TFTP Server IP Address and the Boot Script Name. This functionality is enabled by default but may be disabled.

- If the DHCP Offer or DHCP Ack 'siaddr' field is valid then it is a possible TFTP Server IP Address.
- If the DHCP Offer or DHCP Ack 'file' field is valid then it is a possible Boot File Name. If the DHCP Offer or DHCP Ack Bootfile Name option is present then it is a possible Boot File Name. This may occur when the option Overload / 'file' is asserted in the DHCP Offer or DHCP Ack.

The Boot File is loaded into memory and then executed using the 'autoscr' U-Boot command. However, note the following constraints:

- An override TFTP Server IP Address may be specified in non-volatile memory
- An override Boot Script File Name may be specified in non-volatile memory
- When the TFTP Server IP Address (siaddr) is not valid, the default TFTP Server IP Address is the DHCP Server IP address
- The Boot Script is constructed using the mkimage tool in the CDK under `bootloader/u-boot/tools/mkimage`
- The Boot Script is loaded into a memory location that may be specified
- Optionally, the SIPI/PFI/SFI/LABI application activation methodology may follow boot script execution (Default = Enable)

4.4.9 Automated UA Executable Load and Boot

Optionally, U-Boot supports loading the automated UA executable image into DDR3 SDRAM. U-Boot uses TFTP or the PCIe interface to transfer an image into local Flash memory. U-Boot boots the specified core(s) as soon as the application executable is transferred into DDR3 SDRAM.

U-Boot supports the following application executable:

- OCTEON Simple Executive-Based ELF image
- Linux-Based ELF image
- Generic ELF image

U-Boot automated UA executable load supports both:

- Unique image loading for each cnMIPS 64 core
- Single executive loading for multiple cores

The automated UA executable load is controlled by the configuration for each application executable. Up to 16 application executable entries can reside in the load table, By default, two entries are defined in Table 4-5. Within this table, primary and secondary file names are user-defined.

When the application executable is transferred completely into DDR3 SDRAM, U-Boot boots the specified cores.



NOTE

Core 0 is used for the execution of U-Boot only. Therefore, specify Core 0 as the last core in the load sequence. Subsequent loading is unavailable.

Table 4-5 Load and Boot Image Default Entries

Load and Boot Image Entries	Data Plane Image	Controller Image
Primary Location ID	TFTP Server	TFTP Server
Primary TFTP Ethernet Port ID	Port 0, GbE	Port 0, GbE
Primary TFTP server IP address	DHCP Server IP Address	DHCP Server IP Address
Primary TFTP retry frequency	5 seconds	5 seconds
Primary TFTP number of retries	3 times	3 times
Primary file name	<user-defined>	<user-defined>
Primary file type	Octeon Simple Executive	Linux-based
Secondary location ID	Flash	Flash
Secondary TFTP Ethernet Port ID	Not applicable (N/A)	N/A
Secondary TFTP server IP address	N/A	N/A
Secondary TFTP retry frequency	N/A	N/A
Secondary TFTP number of retries	N/A	N/A
Secondary file name	<user-defined>	<user-defined>
Secondary file type	OCTEON Simple Executive	Linux-based
DDR3 RAM Address Image Destination	0x7000000	0x7000000
DDR3 RAM Address Boot Location	0x7000000	0x7000000
Core Asset Mask	0xFFFFE — 10 cores [number limited to (max_cores - 1) of part]	0x0001 – Core 0

4.4.10 EEPROM Mapping and Tuples

The 64 KB serial EEPROM (hereafter referred to as EEPROM) provides non-volatile data storage for on-board specific hardware configuration information. After reset, U-Boot moves the software image to RAM and then transfers execution to RAM. From RAM, it conducts all subsequent operations. The content of the EEPROM dictates these operations. The EEPROM stores and supports the extended EEPROM tuples listed in Table 4-6.

EEPROM Mapping

The EEPROM provides three storage areas as show in Figure 4-3.

Figure 4-3 EEPROM Mapping EEPROM Tuples

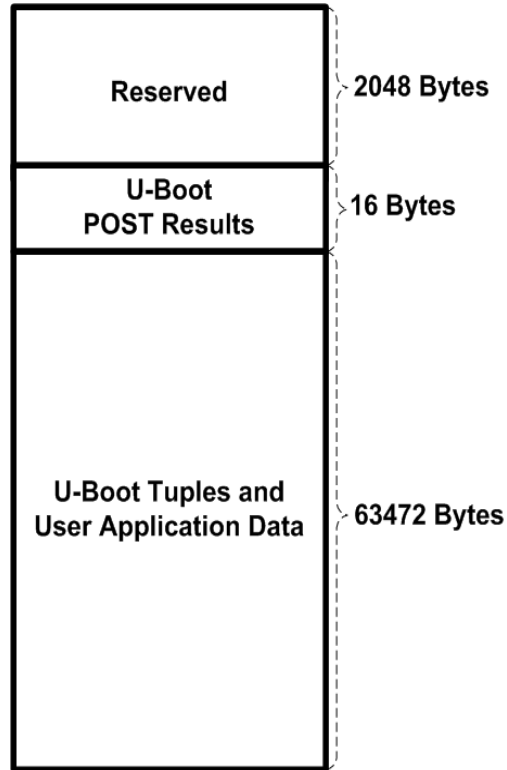


Table 4-6 summarized the EEPROM tuples for the WANic-66512 U-Boot. These tuples supplement the tuple definitions to U-Boot commands.

Table 4-6 EEPROM Tuples

Tuple	Maximum	Description
CSUMTESTI	Up to four	Validates the Checksum test
LABI	Up to 16	Loads and boots information definitions
MEMTEST	Up to four	Creates a Memory test
TESTI	One	Creates a test
PFI	Up to 16	Obtains Primary File Information
RLDRAMT1	One	Creates an low latency memory RLDRAM test
SCRIPT	Up to four	Creates and updates a script
SCRIPTACTIVE	One	Indicates the active script
SERDESTEST1	One	Creates a SerDes test
SFI	Up to 16	Obtains Secondary File Information
SIPI	Up to 10	Defines the Source IP address
UARTCONI	One	Changes the UART console data rate

Tuple Format

The WANic-66512 U-Boot uses the existing U-Boot storage format with a total tuple size of 128 bytes. The tuple consists of both a header and body. The header is defined as follows:

```
typedef struct {
    uint16_t type;
    uint16_t length;
    uint8_t minor_version; /*Minor version increment indicates backward compatible change*/
    uint8_t major_version; /*Major version must be changed when incompatible change made*/
    uint16_t checksum;
    octeon_eeprom_header_t;
```

The body of the tuple is available exclusively for the unique attributes of the specific tuple. For example, the LABI tuple is formed as follows:

```
/****** Load And Boot Information *****/
#define BOOT_COMMAND_MAX_LEN          32
#define IMAGE_ADDR_MAX_LEN           20
#define CORE_ASSERT_MASK_MAX_LEN     8
struct octeon_eeprom_load_and_boot_info_v1
octeon_eeprom_header_t header;
uint8_t boot_command[BOOT_COMMAND_MAX_LEN]; /* Boot Command */
uint8_t image_addr[IMAGE_ADDR_MAX_LEN]; /* Image Address */
uint8_t core_assert_mask[CORE_ASSERT_MASK_MAX_LEN];
/* Core Assert Mask */

};
#define OCTEON_EEPROM_LOAD_AND_BOOT_INFO_VER 1
typedef struct octeon_eeprom_load_and_boot_info_v1
octeon_eeprom_load_and_boot_info_t
```

Displaying a Tuple Address

To display the address of a tuple, enter the command:

```
# tlv_eeprom display
```

The address of all the tuples display similar to the following:

```
=====
MAC_ADDR_TYPE (0x4) tuple found: at addr 0x810
type: 0x4, len: 0x10, csum: 0x1d7, maj_ver: 1, min_ver: 0
MAC base: 00:e0:48:26:01:6f, count: 4
=====
BOARD_DESC_TYPE (0x2) tuple found: at addr 0x820
type: 0x2, len: 0x24, csum: 0x420, maj_ver: 1, min_ver: 0
Board type: W66xx (0x4E26)
Board revision major:1, minor:0
Chip type (deprecated): Unsupported Chip (0x16f)
Chip revision (deprecated) major:4, minor:0
Board ser #: unknown
=====
UBOOT_NORMAL_INFO_TYPE (0x53) tuple found: at addr 0x894
type: 0x53, len: 0x50, csum: 0x8ff, maj_ver: 2, min_ver: 0
Version Number: U-Boot GEF-3.0.1.v/SDK1.8.1-294
This U-Boot Version Is Active
FLASH Partition: UBOOT
  FLASH Offset(Bytes): 0x00200000(2048KBytes)
  FLASH Length(Bytes): 0x001c0000(1792KBytes)
FLASH Partition: Environment
  FLASH Offset(Bytes): 0x003c0000(3840KBytes)
  FLASH Length(Bytes): 0x00020000(128KBytes)
FLASH Partition: NFI(Normal File Information)
  FLASH Offset(Bytes): 0x003e0000(3968KBytes)
  FLASH Length(Bytes): 0x00020000(128KBytes)
=====
```

Deleting a Tuple

To delete a tuple, obtain the address of the tuple then enter the following command:

```
# tlv-eeprom delete <address>
```

Descriptions of WANic-66512 EEPROM tuples follow.

Checksum Validation

Name	CSUMTESTI												
Prototype	<code>tlv_eeeprom set csumtesti [id] [start] [end] [csum location] [csum algorithm][failure action]</code>												
Parameters	<table><tr><td><i>id</i></td><td>Identifies which region to check, where: 0 = First 1 = Second 2 = Third 3 = Fourth</td></tr><tr><td><i>start</i></td><td>Starting address</td></tr><tr><td><i>end</i></td><td>Ending address</td></tr><tr><td><i>csum location</i></td><td>Checksum location</td></tr><tr><td><i>csum algorithm</i></td><td>Checksum algorithm, where: 0 = 8-bit checksum calculation with an 8-bit resultant 1 = 16-bit checksum calculation with a 16-bit resultant 2 = 32-bit checksum calculation with a 32-bit resultant 3 = 64-bit checksum calculation with a 64-bit resultant</td></tr><tr><td><i>failure action</i></td><td>Action that caused the failure, where: 0 = Console error message and continue as normal 1 = Console error message and system hangs</td></tr></table>	<i>id</i>	Identifies which region to check, where: 0 = First 1 = Second 2 = Third 3 = Fourth	<i>start</i>	Starting address	<i>end</i>	Ending address	<i>csum location</i>	Checksum location	<i>csum algorithm</i>	Checksum algorithm, where: 0 = 8-bit checksum calculation with an 8-bit resultant 1 = 16-bit checksum calculation with a 16-bit resultant 2 = 32-bit checksum calculation with a 32-bit resultant 3 = 64-bit checksum calculation with a 64-bit resultant	<i>failure action</i>	Action that caused the failure, where: 0 = Console error message and continue as normal 1 = Console error message and system hangs
<i>id</i>	Identifies which region to check, where: 0 = First 1 = Second 2 = Third 3 = Fourth												
<i>start</i>	Starting address												
<i>end</i>	Ending address												
<i>csum location</i>	Checksum location												
<i>csum algorithm</i>	Checksum algorithm, where: 0 = 8-bit checksum calculation with an 8-bit resultant 1 = 16-bit checksum calculation with a 16-bit resultant 2 = 32-bit checksum calculation with a 32-bit resultant 3 = 64-bit checksum calculation with a 64-bit resultant												
<i>failure action</i>	Action that caused the failure, where: 0 = Console error message and continue as normal 1 = Console error message and system hangs												
Description	Validates the checksum and permits up to four regions of memory to be specified for checksum validation with each region containing a specific checksum. The [id] parameter specifies the region to check.												
Example	<p>The following example illustrates a checksum validation region with:</p> <ul style="list-style-type: none">• A starting address of 0xbfc00000.• An ending address of 0xbfc2fff7.• A checksum location of 0xbfc2fff8.• A checksum algorithm of 3.• A failure action of 0. <pre>tlv_eeeprom set csumtesti 0 0xbfc00000 0xbfc2fff7 0xbfc2fff8 3 0</pre>												

Load and Boot Information

Name	LABI								
Prototype	<code>tlv_eeeprom set labi [id] [boot command] [image address] [command arguments]</code>								
Parameters	<table><tr><td><i>id</i></td><td>Identifier</td></tr><tr><td><i>boot command</i></td><td>Boots an ELF image, where: <i>bootoctlinux</i> = Boots Linux ELF image <i>bootoct</i> = Boots the OCTEON Executive ELF image <i>bootelf</i> = Boots a generic ELF image</td></tr><tr><td><i>image address</i></td><td>Address where the image resides in RAM</td></tr><tr><td><i>command arguments</i></td><td>Input arguments</td></tr></table>	<i>id</i>	Identifier	<i>boot command</i>	Boots an ELF image, where: <i>bootoctlinux</i> = Boots Linux ELF image <i>bootoct</i> = Boots the OCTEON Executive ELF image <i>bootelf</i> = Boots a generic ELF image	<i>image address</i>	Address where the image resides in RAM	<i>command arguments</i>	Input arguments
<i>id</i>	Identifier								
<i>boot command</i>	Boots an ELF image, where: <i>bootoctlinux</i> = Boots Linux ELF image <i>bootoct</i> = Boots the OCTEON Executive ELF image <i>bootelf</i> = Boots a generic ELF image								
<i>image address</i>	Address where the image resides in RAM								
<i>command arguments</i>	Input arguments								
Description	Loads and boots information definition. The LABI tuple is stored in serial EEPROM. A LABI tuple is permitted for each of the Processor cores. The LABI is referenced when the associated PFI and SFI results in an application resident in memory, which is ready for activation on one or more Processor cores.								



NOTE

Enter a dash to indicate values for parameters that are not necessary.

Example The following example configures the LABI 0 for booting an application.

```
tlv_eeeprom set labi 0 bootoctlinux 21000000 coremask=ff00
```

In the above example:

- The `bootoctlinux` command boots the Linux Elf image.
- The image is located at address `21000000` (hex).
- The core mask directs U-Boot to use this image to boot cores 8-15. (Other arguments can be specified along with the coremask.)
- U-Boot terminates once Core 0 is loaded and booted with an executable image.

Memory Testing

Name	MEMTESTI												
Prototype	<code>tlv_eeeprom set memtesti [id] [start] [end] [csum location] [csum algorithm][failure action]</code>												
Parameters	<table><tr><td><i>id</i></td><td>Identifies which region to check, where: 0 = First 1 = Second 2 = Third 3 = Fourth</td></tr><tr><td><i>start</i></td><td>Starting address</td></tr><tr><td><i>end</i></td><td>Ending address</td></tr><tr><td><i>csum location</i></td><td>Checksum location</td></tr><tr><td><i>csum algorithm</i></td><td>Checksum algorithm, where: 0 = 8-bit checksum calculation with an 8-bit resultant 1 = 16-bit checksum calculation with a 16-bit resultant 2 = 32-bit checksum calculation with a 32-bit resultant 3 = 64-bit checksum calculation with a 64-bit resultant</td></tr><tr><td><i>failure action</i></td><td>Action that caused the failure, where: 0 = Console error message and continue as normal 1 = Console error message and system hangs</td></tr></table>	<i>id</i>	Identifies which region to check, where: 0 = First 1 = Second 2 = Third 3 = Fourth	<i>start</i>	Starting address	<i>end</i>	Ending address	<i>csum location</i>	Checksum location	<i>csum algorithm</i>	Checksum algorithm, where: 0 = 8-bit checksum calculation with an 8-bit resultant 1 = 16-bit checksum calculation with a 16-bit resultant 2 = 32-bit checksum calculation with a 32-bit resultant 3 = 64-bit checksum calculation with a 64-bit resultant	<i>failure action</i>	Action that caused the failure, where: 0 = Console error message and continue as normal 1 = Console error message and system hangs
<i>id</i>	Identifies which region to check, where: 0 = First 1 = Second 2 = Third 3 = Fourth												
<i>start</i>	Starting address												
<i>end</i>	Ending address												
<i>csum location</i>	Checksum location												
<i>csum algorithm</i>	Checksum algorithm, where: 0 = 8-bit checksum calculation with an 8-bit resultant 1 = 16-bit checksum calculation with a 16-bit resultant 2 = 32-bit checksum calculation with a 32-bit resultant 3 = 64-bit checksum calculation with a 64-bit resultant												
<i>failure action</i>	Action that caused the failure, where: 0 = Console error message and continue as normal 1 = Console error message and system hangs												
Description	Validates the checksum and permits up to four regions of memory to be specified for checksum validation with each region containing a specific checksum. The [id] parameter specifies the region to check.												
Example	<p>The following example illustrates a checksum validation region with:</p> <ul style="list-style-type: none">• A starting address of 0xBFC00000.• An ending address of 0xBFC2FFF7.• A checksum location of 0xBFC2FFF8.• A checksum algorithm of 3.• A failure action of 0. <pre>tlv_eeeprom set memtesti 0 0xbfc00000 0xbfc2fff7 0xbfc2fff8 3 0</pre>												

PHY Testing

Name	PHYTESTI						
Prototype	<code>tlv_eeprom set phytesti [algorithm][device sel map][failure action]</code>						
Parameters	<table><tr><td><i>algorithm</i></td><td>Algorithm, where the PHY presence detection test is: 0 = Enable 1 = Disable</td></tr><tr><td><i>device sel map</i></td><td>Device selection map is a bit map with the least significant bit corresponding to 0.</td></tr><tr><td><i>failure action</i></td><td>Action that caused the failure, where: 0 = Console error message and continue as normal 1 = Console error message and system hangs</td></tr></table>	<i>algorithm</i>	Algorithm, where the PHY presence detection test is: 0 = Enable 1 = Disable	<i>device sel map</i>	Device selection map is a bit map with the least significant bit corresponding to 0.	<i>failure action</i>	Action that caused the failure, where: 0 = Console error message and continue as normal 1 = Console error message and system hangs
<i>algorithm</i>	Algorithm, where the PHY presence detection test is: 0 = Enable 1 = Disable						
<i>device sel map</i>	Device selection map is a bit map with the least significant bit corresponding to 0.						
<i>failure action</i>	Action that caused the failure, where: 0 = Console error message and continue as normal 1 = Console error message and system hangs						
Description	Creates a test function for the PHY by detecting the presence and returning the status of the test.						
Example	<p>The following example configures an eight test that executes on ports 0-7.</p> <pre>tlv_eeprom set phytesti 0 0x00ff 0</pre> <p>where:</p> <ul style="list-style-type: none">• Algorithm 0 supports a presence detection test.• The device selection map is a bit map with the least significant bit corresponding to 0.						

PMEM Testing

Name PMEMTESTI

Prototype `tlv_eeeprom set pmemtesti [algorithm] [verbose] [failure action]`

Parameters

<i>algorithm</i>	Algorithm, where the PMEM presence detection test is: 0 = Enable 1 = Disable
<i>verbose</i>	Additional test information, where: 0 = Disable 1 = Enable
<i>failure action</i>	Action that caused the failure, where: 0 = Console error message and continue as normal 1 = Console error message and system hangs

Description Creates a test function for the PMEM.



NOTE

PMEMTESTI testing only tests PMEM and *excludes* system memory. See MEMTESTI for information on testing system memory.

Example The following example configures a PMEM test.

```
tlv_eeeprom set pmemtesti 0 0 0
```

Primary File Information

Name	PFI														
Prototype	<code>tlv_eeeprom set pfi [id] [flash floc] [flash fsize] [SIPI ID] [TFTP fname] [TFTP serve IP addr] [TFTP num retries]</code>														
Parameters	<table><tr><td><i>id</i></td><td>Identifier</td></tr><tr><td><i>flash floc</i></td><td>Flash file location</td></tr><tr><td><i>flash fsize</i></td><td>Flash file size</td></tr><tr><td><i>SIPI ID</i></td><td>Source IP address</td></tr><tr><td><i>TFTP fname</i></td><td>TFTP filename</td></tr><tr><td><i>TFTP serve IP addr</i></td><td>TFTP server IP address</td></tr><tr><td><i>TFTP num retries</i></td><td>Number of retries for TFTP</td></tr></table>	<i>id</i>	Identifier	<i>flash floc</i>	Flash file location	<i>flash fsize</i>	Flash file size	<i>SIPI ID</i>	Source IP address	<i>TFTP fname</i>	TFTP filename	<i>TFTP serve IP addr</i>	TFTP server IP address	<i>TFTP num retries</i>	Number of retries for TFTP
<i>id</i>	Identifier														
<i>flash floc</i>	Flash file location														
<i>flash fsize</i>	Flash file size														
<i>SIPI ID</i>	Source IP address														
<i>TFTP fname</i>	TFTP filename														
<i>TFTP serve IP addr</i>	TFTP server IP address														
<i>TFTP num retries</i>	Number of retries for TFTP														
Description	Uses the TFTP to obtain the PFI definition from an TFTP server, or to use a Flash file. A PFI tuple is permitted for each core of the Processor (up to 12).														



NOTE

Enter a dash (–) to indicate values that are not required.

Example The following example configures the PFI to acquire an application from a TFTP server.

```
tlv_eeeprom set pfi 0--0 vmlinux.64 10.71.1.11 4
```

In this example:

- PFI is 0 if configured.
- A dash specifies the Flash file location and Flash File size since they are not needed.
- SIPI0 is the source IP address information.
- The TFTP server is specified as 10.71.1.11.
- The number of retries is 4.

RLDRAM Testing

Name	RLDRAMTI								
Prototype	<code>tlv_eeeprom set rldramti [address] [datasize] [replication type] [display option]</code>								
Parameters	<table><tr><td><code>address</code></td><td>Address</td></tr><tr><td><code>datasize</code></td><td>Size of the data</td></tr><tr><td><code>replication type</code></td><td>Word replication factor</td></tr><tr><td><code>display option</code></td><td>Displays to the screen option</td></tr></table>	<code>address</code>	Address	<code>datasize</code>	Size of the data	<code>replication type</code>	Word replication factor	<code>display option</code>	Displays to the screen option
<code>address</code>	Address								
<code>datasize</code>	Size of the data								
<code>replication type</code>	Word replication factor								
<code>display option</code>	Displays to the screen option								
Description	Creates a test function for the RLDRAM.								
Example	<p>The following example tests the RLDRAM:</p> <pre>tlv_eeeprom set rldramti 0 1024 0</pre> <p>In this example:</p> <ul style="list-style-type: none">• The data size is 1024.• The replication type is 0.								

SerDes Testing

Name	SERDESTESTI				
Prototype	<code>tlv_eeeprom set serdestesti [<i>algorithm</i>] [<i>failure action</i>]</code>				
Parameters	<table><tr><td><i>algorithm</i></td><td>Algorithm</td></tr><tr><td><i>failure action</i></td><td>Action that caused the failure, where: 0 = Console error message and continue as normal 1 = Console error message and system hangs</td></tr></table>	<i>algorithm</i>	Algorithm	<i>failure action</i>	Action that caused the failure, where: 0 = Console error message and continue as normal 1 = Console error message and system hangs
<i>algorithm</i>	Algorithm				
<i>failure action</i>	Action that caused the failure, where: 0 = Console error message and continue as normal 1 = Console error message and system hangs				
Description	Creates a test function for the SerDes by detecting the SerDes presence and returning the status of the test.				
Example	<p>The following example configures a SerDes test.</p> <pre>tlv_eeeprom set serdestesti 0 0</pre> <p>In this example:</p> <ul style="list-style-type: none">• The algorithm is 0.• The failure action is 0.				

Script

Name	SCRIPT				
Prototype	<code>tlv_eeprom set script [id] [script command] { }{script command}.....</code>				
Parameters	<table><tr><td><i>id</i></td><td>Script ID</td></tr><tr><td><i>script command</i></td><td>Script command</td></tr></table>	<i>id</i>	Script ID	<i>script command</i>	Script command
<i>id</i>	Script ID				
<i>script command</i>	Script command				
Description	Creates and updates a sequence of U-Boot commands to execute during initialization. The WANic-66512 provides storage for up to four scripts; however, only one active script is permitted at any given time. See the <code>scriptactive</code> function for more information on creating an active script.				
Example	<p>The following example creates a script tuple with two commands:</p> <pre>tlv_eeprom set script 0 dhcp setenv serverip 10.71.1.11</pre> <p>Add subsequent commands to this script tuple as follows:</p> <pre>tlv_eeprom set script 0 tftpboot 21000000 vmlinux.64</pre>				

Script Active

Name	SCRIPTACTIVE
Prototype	<code>tlv_eeeprom set scriptactive [id]</code>
Parameters	<i>id</i> Script ID
Description	Specifies the active script.
Example	<p>The following example creates a <code>Scriptactive</code> tuple with Script 0 as the active script:</p> <pre>tlv_eeeprom set scriptactive 0</pre> <p>The WANic-66512 executes the following commands from Script 0 during initialization:</p> <pre>dhcp setenv serverip 10.71.1.11 tftpboot 21000000 vmlinux.64</pre>



NOTE

See the `SCRIPT` tuple for more information on creating the script used in this example.

Source IP Address Information

Name SIPI

Prototype `tlv_eeprom set sipi [id] [ethernet portid] [DHCP enable] [DHCP num retries] [IP addr] [subnet mask] [default GW IP addr];`

Parameters

<i>id</i>	Identifier
<i>ethernet portid</i>	Ethernet Port ID
<i>DHCP enable</i>	Enables or disables DHCP, where: 0 = Disable 1 = Enable
<i>DHCP num retires</i>	Number of retires
<i>IP address</i>	Internet Protocol address
<i>subnet mask</i>	Subnet mask
<i>default GW IP addr</i>	Default gateway internet protocol address

Description Defines the source IP address information. A SIPI tuple is permitted for each WANic-66512 Ethernet interface.



NOTE

The IP address, subnet mask, and default gateway are received from the DHCP server when DHCP is enabled.

Example The following example configures SIPI 0 for Ethernet 0 with DHCP enabled. The IP address, subnet mask, and default gateway are received from the DHCP server and are not entered as part of SIPI 0.

```
tlv_eeprom set sipi 0 octeth0 1 0---
```

The following example, configures a static IP address:

```
tlv_eeprom set sipi 0 octeth0 0 0 10.72.1.21 255.255.255.0.0 10.72.1.1
```

In the above example:

- The ID is 0.
- The Ethernet port ID is `octeth0`.
- DCHP is enabled.
- The IP address is `10.72.1.21`.
- The subnet mask is `255.255.255.0`.
- The default gateway IP address is `10.72.1.1`.

Secondary File Information

Name	SFI														
Prototype	<code>tlv_eeeprom set sfi [id] [flash floc] [flash fsize] [SIPI ID] [TFTP fname] [TFTP serve IP addr] [TFTP num retries]</code>														
Parameters	<table><tr><td><i>id</i></td><td>Identifier</td></tr><tr><td><i>flash floc</i></td><td>Flash file location</td></tr><tr><td><i>flash fsize</i></td><td>Flash file size</td></tr><tr><td><i>SIPI ID</i></td><td>Source IP identifier</td></tr><tr><td><i>TFTP fname</i></td><td>TFTP file name</td></tr><tr><td><i>TFTP serve IP addr</i></td><td>TFTP server IP address</td></tr><tr><td><i>TFTP num retries</i></td><td>Number of retries for the TFTP</td></tr></table>	<i>id</i>	Identifier	<i>flash floc</i>	Flash file location	<i>flash fsize</i>	Flash file size	<i>SIPI ID</i>	Source IP identifier	<i>TFTP fname</i>	TFTP file name	<i>TFTP serve IP addr</i>	TFTP server IP address	<i>TFTP num retries</i>	Number of retries for the TFTP
<i>id</i>	Identifier														
<i>flash floc</i>	Flash file location														
<i>flash fsize</i>	Flash file size														
<i>SIPI ID</i>	Source IP identifier														
<i>TFTP fname</i>	TFTP file name														
<i>TFTP serve IP addr</i>	TFTP server IP address														
<i>TFTP num retries</i>	Number of retries for the TFTP														
Description	Creates the SFI definition. A SFI tuple is permitted for each core of the Multi-Core Processor for up to 10 cores. The SFI is referenced when either the associated PFI is not defined, or the TFTP transfer associated with the PFI encounters an error condition.														
Example	<p>The following example configures SFI 0 for acquiring an allocation from Flash.</p> <pre>tlv_eeeprom set sfi 0 bfc80000 80000 0--0</pre> <p>In the above example:</p> <ul style="list-style-type: none">• SFI 0 is configured.• The Flash-based application file address must be specified (<code>bfc80000</code>).• The Flash-based application file size is specified (<code>80000</code>) because the application is to be transferred to RAM.• SIPI 0 must be specified even though it is not for Flash-based operations.• A dash is specified for both the TFTP file name and TFTP server IP address as this information is not required.• The number of retries is <code>0</code>.														

UART Configuration

Name	UARTCONI
Prototype	<code>tlv_eeprom set uartconi [<i>baudrate</i>])</code>
Parameters	<i>baudrate</i> Baud rate
Description	Changes the console UART data rate by creating the UARTCONI tuple in non-volatile storage.
Example	The following example configures the console UART data rate for 115200 bps. <code>tlv_eeprom set uartconi 115200</code>

UART MMC Interface

Name	UARTMMCI
Prototype	<code>tlv_eeeprom set uartmmci [<i>baudrate</i>]</code>
Parameters	<i>baudrate</i> Baud rate
Description	Changes the MMC UART data rate by creating the UARTMMCI tuple in non-volatile storage.
Example	The following example configures the MMC UART data rate for 112500bps. <code>tlv_eeeprom set uartmmci 112500</code>

U-Boot Normal Image Version

Name uboot_normal_info

Description= Contains the Normal U-Boot image information stored in non-volatile storage.

Example UBOOT_NORMAL_INFO_TYPE (0x53) tuple found: at addr 0x834
type: 0x53, len: 0x50, csum: 0x85b, maj_ver:2, min_ver:0
Version Number: U-Boot GEF-3.2.7/SDK2.0.0-366
This U-Boot Version Is Active
Flash Partition: UBOOT
 FLASH Offset (Bytes): 0x00200000(2048KBytes)
 FLASH Length (Bytes): 0x00020000(1792KBytes)
Flash Partition: Environment
 FLASH Offset (Bytes): 0x003c0000(3840KBytes)
 FLASH Length (Bytes): 0x00020000(128KBytes)
FLASH Partition: NFI (Normal File Information)
 FLASH Offset (Bytes): 0x003e0000(3968KBytes)
 FLASH Length (Bytes): 0x00020000(128KBytes)

4.4.11 EEPROM Tuple Update and Enumeration

The WANic-66512 U-Boot updates the enumerated EEPROM tuple types as follows:

```
enum eeprom_types_enum {
    EEPROM_NULL_TYPE = 0,
    EEPROM_CLOCK_DESC_TYPE,
    EEPROM_BOARD_DESC_TYPE,
    EEPROM_CHIP_CAPABILITY_TYPE,
    EEPROM_MAC_ADDR_TYPE,
    EEPROM_VOLT_MULT_TYPE,
    EEPROM_NIC_XL_DESC_TYPE,
#ifdef CONFIG_OCTEON_w66xx
/* Source IP Information */
    EEPROM_SIPI0_TYPE,
    EEPROM_SIPI1_TYPE,
    EEPROM_SIPI2_TYPE,
    EEPROM_SIPI3_TYPE,
    EEPROM_SIPI4_TYPE,
    EEPROM_SIPI5_TYPE,
    EEPROM_SIPI6_TYPE,
    EEPROM_SIPI7_TYPE,
    EEPROM_SIPI8_TYPE,
    EEPROM_SIPI9_TYPE,
/* Primary File Information */
    EEPROM_PFI0_IMAGE_TYPE,
    EEPROM_PFI1_IMAGE_TYPE,
    EEPROM_PFI2_IMAGE_TYPE,
    EEPROM_PFI3_IMAGE_TYPE,
    EEPROM_PFI4_IMAGE_TYPE,
    EEPROM_PFI5_IMAGE_TYPE,
    EEPROM_PFI6_IMAGE_TYPE,
    EEPROM_PFI7_IMAGE_TYPE,
    EEPROM_PFI8_IMAGE_TYPE,
    EEPROM_PFI9_IMAGE_TYPE,
    EEPROM_PFI10_IMAGE_TYPE,
    EEPROM_PFI11_IMAGE_TYPE,
    EEPROM_PFI12_IMAGE_TYPE,
    EEPROM_PFI13_IMAGE_TYPE,
    EEPROM_PFI14_IMAGE_TYPE,
    EEPROM_PFI15_IMAGE_TYPE,
/* Secondary File Information */
    EEPROM_SFI0_IMAGE_TYPE,
    EEPROM_SFI1_IMAGE_TYPE,
    EEPROM_SFI2_IMAGE_TYPE,
    EEPROM_SFI3_IMAGE_TYPE,
    EEPROM_SFI4_IMAGE_TYPE,
    EEPROM_SFI5_IMAGE_TYPE,
    EEPROM_SFI6_IMAGE_TYPE,
    EEPROM_SFI7_IMAGE_TYPE,
    EEPROM_SFI8_IMAGE_TYPE,
    EEPROM_SFI9_IMAGE_TYPE,
    EEPROM_SFI10_IMAGE_TYPE,
    EEPROM_SFI11_IMAGE_TYPE,
    EEPROM_SFI12_IMAGE_TYPE,
    EEPROM_SFI13_IMAGE_TYPE,
    EEPROM_SFI14_IMAGE_TYPE,
    EEPROM_SFI15_IMAGE_TYPE,
```

```

/* Load And Boot Information */
    EEPROM_LABI0_IMAGE_TYPE,
    EEPROM_LABI1_IMAGE_TYPE,
    EEPROM_LABI2_IMAGE_TYPE,
    EEPROM_LABI3_IMAGE_TYPE,
    EEPROM_LABI4_IMAGE_TYPE,
    EEPROM_LABI5_IMAGE_TYPE,
    EEPROM_LABI6_IMAGE_TYPE,
    EEPROM_LABI7_IMAGE_TYPE,
    EEPROM_LABI8_IMAGE_TYPE,
    EEPROM_LABI9_IMAGE_TYPE,
    EEPROM_LABI10_IMAGE_TYPE,
    EEPROM_LABI11_IMAGE_TYPE,
    EEPROM_LABI12_IMAGE_TYPE,
    EEPROM_LABI13_IMAGE_TYPE,
    EEPROM_LABI14_IMAGE_TYPE,
    EEPROM_LABI15_IMAGE_TYPE,
/* Script ID Active */
    EEPROM_SCRIPT_ID_ACTIVE_TYPE,
                                /* Script ID Active */
/* Script 0 */
    EEPROM_SCRIPT0_TYPE, /* Script */
/* Script 1 */
    EEPROM_SCRIPT1_TYPE, /* Script */
/* Script 2 */
    EEPROM_SCRIPT2_TYPE, /* Script */
/* Script 3 */
    EEPROM_SCRIPT3_TYPE, /* Script */
/* POST Information */
    EEPROM_CSUMTESTI0_TYPE,
    EEPROM_CSUMTESTI1_TYPE,
    EEPROM_CSUMTESTI2_TYPE,
    EEPROM_CSUMTESTI3_TYPE,
    EEPROM_MEMTESTI0_TYPE,
    EEPROM_MEMTESTI1_TYPE,
    EEPROM_MEMTESTI2_TYPE,
    EEPROM_MEMTESTI3_TYPE,
    EEPROM_TESTI_TYPE,
    EEPROM_RLDRAMTI_TYPE,
    EEPROM_SERDESTESTI_TYPE,
/* UART-Console Information */
    EEPROM_UARTCONI_TYPE,
/* UART-MMC Information */
    EEPROM_UARTMMCI_TYPE,
#endif /* CONFIG_OCTEON_wnpa38xx */
    EEPROM_MAX_TYPE,
/* Start of range (inclusive) for customer use */
    EEPROM_CUSTOMER_RESERVED_START = 0xf000,
                                /* End of range (inclusive) for customer use */
    EEPROM_CUSTOMER_RESERVED_END = 0xff00,
    EEPROM_END_TYPE = 0xffff
};

```

4.5 Linux Support Packages

The Linux Support Package (LSP) and Wind River Linux PNE-LE Board Support Package (BSP) provide a suite of functions and tools for accessing and using the WANic-66512 hardware.

The LSP and BSP supply configuration and functional drivers for devices that are not supported by the Linux Open Source community. The LSP and BSP provide drivers necessary to control all I/O interfaces.

The LSP and BSP provide hardware address definitions and important register bit mask definitions. These support packages permit the mapping of every device into kernel address space for configuration and register access. They also permit fault handling and state logging. In addition, support packages permit devices to be removed from the kernel address space.

The LSP and BSP provide the device configuration drivers to manage register configuration content for each programmable device. This includes register access drivers for reading and for clearing error status from each device that has error status registers. The register access drivers return either *Pass* (0) or *Fail* (negative integer error code) as access completion codes, and access size is specified in the interface.

4.5.1 BSP

The Wind River Linux PNE-LE BSP is provided to support the Wind River Linux PNE-LE operating system. This BSP provides a collection of code, device drivers, and debugging features to successfully use Wind River PNE-LE Linux with the Multi-Core Processor.

4.5.2 LSP

The LSP consists of a driver, diagnostic application, and an In-Service Daemon.



NOTE

Each component of the LSP is covered by the GNU General Public License, version 2. (See "[Appendix B: GNU General Public License V2](#)" for more information.)

- The driver provides the API for all of the accessible hardware devices on the WANic-66512.
- The diagnostic application provides in-service tests as well as U-Boot and Linux upgrade utilities.
- The In-Service Daemon is a background process that monitors the hardware platform responsiveness continuously, and alerts the system if no response is given.

The LSP reduces UA development time by providing an API for facilitating development of WANic-66512 hardware components. It enriches the programming environment by providing configuration, status and testing utilities. The LSP may execute on a MIPS64 core other than Core 0.

The LSP supports the following devices:

- EEPROM
- Flash
- SFPs

The LSP can be configured for PCI Host Mode where the WANic-66512 runs its own Linux kernel.

The LSP contains the following functions, drivers, and APIs to assist in creating the UA:

- Flash Driver
- NPA Driver
 - Linux `/proc/` File System
 - I/O Control (IOCTL) API
- In-Service Daemon
- Diagnostic Application
- EEPROM Tuple Storage API
- EEPROM Error Code Storage and Retrieval API
- TWSI Primitives for I2C
- SFP Configuration and Status Functions API

The LSP provides an API for configuration and status of the front panel SFPs.

The Flash Driver allows information to be read, deleted, and retrieved from Flash.

The NPA Driver provides the API for communication with the on-board hardware. The NPA Driver also creates the `/proc` file. The `/proc` file provides access to hardware and firmware information. The NPA Driver also contains an IOCTL API, which allows a UA to communicate with the hardware.

The NPA Driver provides functions to read and to write tuples into EEPROM. An EEPROM API allows the storage and retrieval of error codes during diagnostic testing.

The In-Service Daemon is a background process that monitors the hardware platform continuously and reports any faults detected.

The Diagnostic Application provides in-service tests as well as U-Boot and Linux upgrade utilities.

Primitives for the industry standard Two-Wire Serial Interface (TWSI) provide I2C access on the WANic-66512 and support select devices.

4.5.3 Flash Driver

The Flash Driver performs the actual input and output of code from the Flash. Code execution is supported from this Flash during hardware initialization only.

The Flash Driver provides a standard character device that allows information to be read, deleted, and retrieved from the Flash. Typically, this device is identified as `/dev/mtd0`.



NOTE

All paths are relative to the Linux Kernel source.

The Flash Driver functions in Table 4-7 are standard Linux system calls:

- Typically, these functions are defined in `include/sys/ioctl.h` and `include/unistd.h` in the Linux Kernel source.
- Micro `MEMGETINFO` and `MEMERASE` are found in the standard Memory Technology Device (MTD) subsystem for Linux micros and are defined in `include/mtd/mtd-abi.h`.
- Micro `SEEK_SET` is the standard file control micro. This function is defined in `include/fcntl.h`.
- Data structure `mtd_info_user` and `erase_info_user` are also defined in `include/mtd/mtd-abi.h`.

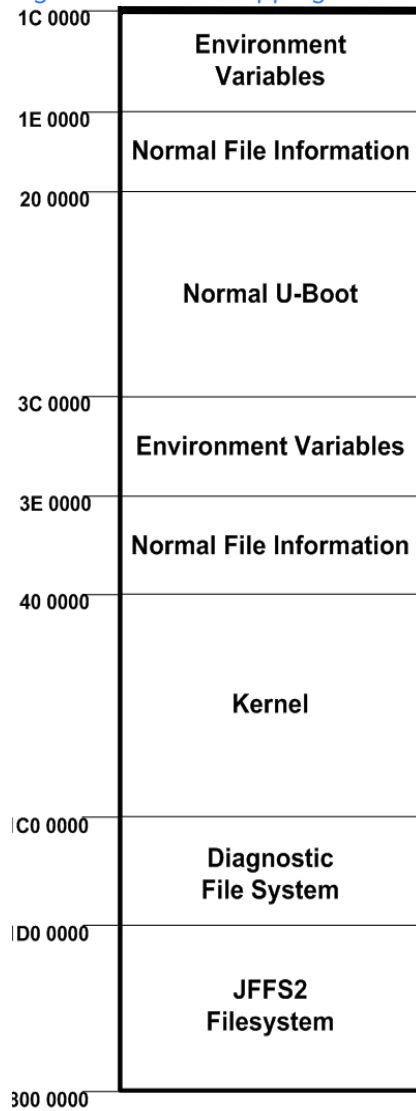
Table 4-7 Flash Driver Functions

Action	Functions
Get Flash Information	<code>int ioctl (int fd, MEMGETINFO, struct mtd_info_user * mtc);</code>
Erase	<code>int ioctl (int fd, MEMERASE, struct erase_info_user erase);</code>
Seek	<code>_off_t lseek (int fd, _toffset, SEEK_SET);</code>
Read	<code>ssize_t read(int fd, void * buf, size_t nbytes);</code>
Write	<code>ssize_t (int fd, void * buf, size_t n);</code>

Flash Mapping

The WANic-66512 U-Boot maps the Flash on the Boot Bus/Chip Select 0. The Flash contains 128 MB. The base address is **17C0 0000** (B7C0 0000). Figure 4-4 shows the default factory-installed Flash mapping.

Figure 4-4 Flash Mapping



Accessing the Flash

To access the Flash without creating programming code, use the `mtd_debug` function found in the MTD Utility. The MTD Utility is a collection of tools for various Flash devices found on <http://sourceware.org/jffs2/>.



NOTE

The `mtd_debug` function assumes that the input/output file is in binary format.

Use `mtd_debug` in the following format:

```
mtd_debug info <device>
mtd_debug read <device> <offset> <len> <dest-filename>
mtd_debug <device> <offset> <len> <source-filename>
mtd_debug erase <device> <offset> <len>
```

An example follows:

```
mtd-utils# mtd_debug info /dev/mtd
mtd.type = MTD_NANDFLASH
mtd.flags = MTD_CAP_NANDFLASH
mtd.size = 8388608 (8M)
mtd.erasysize = 8192 (8K) (Note: Must be a multiple of 8)
mtd.size = 512 (Note: Must be a multiple of 512)
mtd.ecctype = MTD_ECC_SW
regions = 0
```

Partitioning the Flash

By default, the Flash is partitioned at the factory as shown in Figure 4-3, and described in Table 4-8.

Table 4-8 Flash Partitioning

Size	Access	Description
128 KB	Read-Only	U-Boot Environment Variables
40 MB	Read/Write	Compressed Kernel
Remaining MB (depending on the configuration)	Read/Write	JFFS2 File System

This partitioning layout can be customized. For example, it can be useful to create two dedicated Flash regions: one for read-only and one is writable.

To customized Flash partitioning, use the `mtdparts` command line option. For example, the following command partitions the Flash into the factory default settings:

```
mtdparts=s_mapped_flash:1408k(bootloader)ro,128k(bootloader_env)ro,40960k(kernel),-(jffs2)
```

To display the Flash partitioning, enter the `cat /proc/mtd` command from the Linux command line. Partitioning for the Flash displays similar to the following:

```
cat /proc/mtd
dev:   size  erasesize name
mtd0: 08000000 00020000 "phys_mapped_flash"
mtd1: 001c0000 00020000 "bootloader"
mtd2: 00020000 00020000 "bootloader_env"
mtd3: 02800000 00020000 "kernel"
mtd4: 00100000 00020000 "diagfs"
mtd5: 05300000 00020000 "jffs2"
```

Reprogramming the Linux Kernel in Flash

The LSP allows reprogramming of the Linux kernel in Flash:

- Automatically
- Manually
- Via the Diagnostic Utility

To reprogram the Linux Kernel in Flash, use one of the following methods:

- To manually re-program the Linux Kernel in Flash, perform the following steps:
 - a. Copy the new U-Boot image and determine the size, where size is X bytes.
 - b. Erase the Flash by entering:

```
mtd_debug erase /dev/mtd2 0 Y
```

where:
 - Y is the smallest number that is greater than X bytes, and Y is also a multiple of 131072.
 - 131072 is the Erase Size of the Flash. This means the specified Erase Size must be a multiple of 131072 when erasing *any part* of the Flash.
 - c. Write the new Linux Kernel image to Flash by entering:

```
mtd_debug write /dev/mtd2 0 <IMAGE> X
```

where:
 - $<IMAGE>$ is the new Linux Kernel image file name including full path
 - X is the size of the new Linux Kernel image
- Use the Diagnostic Utility (npaDiag) Flash Menu option. See “[Chapter 5: Linux Support Package \(LSP\) Applications](#)” for more information.

Reprogramming U-Boot in Flash

The LSP allows you to reprogram the U-Boot Bootloader manually or via the Diagnostic Utility.

To reprogram the U-Boot Bootloader, use one of the following methods:

- To manually reprogram the U-Boot Bootloader, perform the following steps:
 - a. Copy the new U-Boot Bootloader image and determine the size, where size is X bytes.
 - b. Erase the Flash by entering:

```
mtd_debug erase /dev/mtd0 0 Y
```

where:
 - Y is the smallest number that is greater than X bytes, and Y is also a multiple of 131072.
 - 131072 is the Erase Size of the U-Boot Bootloader. This means the specified Erase Size must be a multiple of 131072 when erasing *any part* of the U-Boot Bootloader.
 - c. Write the new U-Boot Bootloader image to Flash by entering:

```
mtd_debug write /dev/mtd0 0 <IMAGE> X
```

where:
 - $<IMAGE>$ is the new Linux Kernel image file name including full path
 - X is the size of the new Linux Kernel image
- Use the Diagnostic Utility (npaDiag) Flash Menu option. See “[Chapter 5: Linux Support Package \(LSP\) Applications](#)” for more information.

Running the Flash File System

To use the Flash file system, run:

```
mount -t jffs2 /dev/mtdblock3 MOUNT_POINT
```

where $MOUNT_POINT$ is the location in which to mount the Flash file system.

Refer to Cavium SDK documentation and <http://sourceware.org/jffs2/> for more information on JFFS2.

This section describes how to update the firmware in Flash using the PCIe bus. It is assumed that the new image has been copied to the host machine *prior* to performing this procedure.

This section contains instructions for updating the firmware over PCIe for SDK Version 2.0.

Perform the following steps to upgrade the Flash over PCIe:

1. Visually check the S1 DIP Switch bank to see if Switch 3 is in the On or Off position.
 - When Switch 3 is On, it is set for booting over PCI Express.
 - When Switch 3 is Off, it is set for booting from Flash.

To perform booting from the PCIe, set the Switch 3 ON.

See [Chapter 2: Hardware Description](#) for the location of the S1 DIP Switch bank.

2. Install the board and power on the system.
3. At the system prompt, enter the following command, (where W66XX and w66xx are case sensitive):

```
./oct-remote-boot -board=W66XX u-boot-octeon_w66xx.bin
```

The U-Boot prompt displays.

4. Enter the following command to load the binary into OCTEON RAM remotely over the PCIe, and to burn the new file into Flash:

```
./oct-remote-load 0 u-boot-octeon_w66xx.bin
```

where 0 tells U-Boot to load the new binary file into the memory location specified in the environment variable `loadaddr`.

5. At the U-Boot prompt, enter the following command to force the Normal U-Boot to be burned into Flash:

```
./oct-remote-bootcmd "run bootloader_flash_update_normal"
```

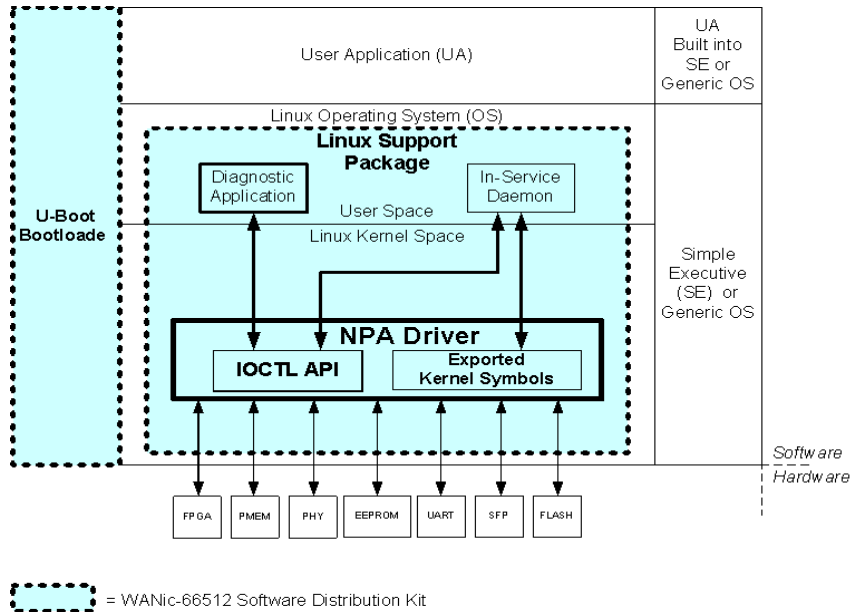
6. Set Switch 3 of the S1 DIP Switch bank to the OFF position.
7. Reset the WANic-66512 to boot the new image out of Flash.

4.5.4 NPA Driver

The NPA Driver is a Linux kernel module that provides the API for communications with the WANic-66512 on-board hardware, and also creates and updates the `/proc` file directory. The `/proc` file directory contains a hierarchy of special files, which represent the current state of the kernel and allows applications and users to peer into the kernel's view of the system.

The NPA Driver contains the IOCTL API and Exported Kernel Symbols as shown in Figure 4-5.

Figure 4-5 NPA Driver



4.5.4.1 Proc File System

The `/proc/` file system provides access to information about the state of the WANic-66512 hardware and software. Access the `/proc/driver/` and the `/proc/net/` directories to obtain information about the following:

- Basic RGMII Ethernet PIP/PKO statistics
- Temperature sensor readings
- U-Boot environment variables
- Named allocated memory space

The `/proc/driver/` directory contains information for specific drivers in use by the kernel. The `/proc/net/` directory contains information about Linux networking.

When the driver is configured for `PCI_HOST_MODE`, enter the following commands:

```
cat /proc/driver/gefes<x>/ethernetstats
cat /proc/driver/gefes<x>/temperature
```

where `x` =device number

The `/environment` `proc` entry is a writable file when setting environment variables from embedded Linux.

Values are added or change when echoed to the file:

```
echo var=val > /proc/drivers/gefes<x>/environment
```

where `x` = device number

When the driver is configured for `PCI_HOST_MODE`, the named memory allocation is listed in the `proc` entry, `named_alloc_list`. Enter the following command:

```
cat /proc/driver/gefes<x>/named_alloc_list
```

where `x` = device number

Proc File Updates

The NPA Driver creates and updates the `/proc` file system for temperature sensor readings, basic RGMII Ethernet PIP/PKO statistics, U-Boot environment variable `named_alloc_list`, and Processor performance monitoring registers.

Commands to generate Proc Ethernet, temperature and U-Boot environments readings configured for `PCI_HOST_MODE` are as follows:

```
/proc/drivers/gefes<x>/ethernetstats
/proc/drivers/gefes<x>/temperature
/proc/drivers/gefes<x>/environment
/proc/drivers/gefes<x>/named_alloc_list
```

where `x` = device number

4.5.4.2 IOCTL API

The WANic-66512 IOCTL API supports commands that allow the UA to interface to the WANic-66512 hardware. The IOCTL API within the NPA Driver supports the following:

- OCTEON Configuration and Status Register (CSR)
- EEPROM tuples
- PHY devices (if available)
- LEDs
- TWSI (I2C access)
- PCIe
- U-Boot environment variables
- Port devices

The IOCTL code contains commands and structures to configure and to monitor all hardware features enabled on the WANic-66512. Figure 4-6 and Table 4-9 list supported IOCTL commands and related structures, which can be sent to the NPA Driver through the IOCTL interface.

Figure 4-6 IOCTL Commands and Structures

```
enum eNpaCommands
{
    _WRITE_TLV_TUPLE = 0, /**<Writes a TUPLE to the EEPROM*/
    _DELETE_TUPLE,      /**<Deletes a TUPLE from the EEPROM*/
    _GET_TLV_TUPLE,     /**<Retrieves TUPLE data from the EEPROM*/
    _GET_NEXT_TUPLE,   /**<Retrieves the next TUPLE while enumerating TUPLE
                        values*/
    _STORE_POST_ERROR, /**<Store a 8 bit POST error code in the EEPROM*/
    _GET_POST_ERROR,   /**<Retrieves all 16 8bit POST error codes from EEPROM*/
    _CLEAR_POST_ERROR, /**<Clears all POST results from the EEPROM*/
    _TWSI_WRITE,       /**<Writes a value to the specified TWSI device*/
    _TWSI_READ,        /**<Reads a value from the specified TWSI device*/
    __WRITE,           /**<Writes a value to the registers*/
    __READ,            /**<Reads a value from the registers*/
    __VS_WRITE,        /**<Writes a value to the SFP registers*/
    __VS_READ,         /**<Reads a value from the SFP registers*/
    _SET_LED_STATE,    /**<Sets the LED state*/
    _GET_LED_STATE,    /**<Retrieves the current LED state*/
    _DEVICE_COUNT,     /**<In PCI Target configuration, returns the number of
                        PCI devices found*/
    _PEEK,             /**<In PCI Target configuration, peek at an address*/
    _POKE,             /**<In PCI Target configuration, poke at an address*/
    _GET_BOARD_INFO,   /**<Retrieves version information for the board*/
    _SET_ENVIRONMENT,  /**<'setenv' for U-Boot/ environment variable*/
    _GET_ENVIRONMENT,  /**<'getenv' for U-Boot/ environment variable*/
    _GET_NEXT_ENVIRONMENT /**<Used to iterate through the environment list*/
    _SET_PORT_ADMIN_STATE, /**<Sets a port state to 'up' or 'down'; allows up
                        on link or forces down on link*/
    _GET_PORT_ADMIN_STATE, /**<Queries the current admin 'up' or 'down' state*/
    _GET_PORT_STATE,    /**<Queries information about the port state*/
    _SET_PORT_STATE,    /**<Queries information about the port state*/
    _GET_PORT_STAT,     /**<Retrieves the statistic at index number specified*/
    _GET_PORT_STAT_COUNT, /**<Retrieves a count of the number of statistics
                        that the port provides*/
    _RESET_PORT_STATS,  /**<Resets the statistic counters*/
    _SET_LOOPBACK,      /**<Sets the loopback state via NPA_LOOPBACK_INFO_t*/
    _GET_LOOPBACK,      /**<Gets the loopback state via NPA_LOOPBACK_INFO_t*/
    _NPA_DEV_EX_CMD = 0xF0 /**<Extended device specific commands*/
};
```


IOCTL Commands

Each of the following enumerated commands translates to a hardware function. Table 4-9 details each IOCTL command and its corresponding hardware functionality.

Table 4-9 IOCTL Command Structures

Command	Structure	Description
<code>_DELETE_TUPLE</code>	<code>NPA_TUPLE_INFO_t</code> Filled structure describes the tuple to delete.	Deletes a tuple from EEPROM.
<code>_GET_TLV_TUPLE</code>	<code>NPA_TUPLE_INFO_t</code> Filled structure describes the tuple to retrieve.	Retrieves tuple data from the EEPROM. If a tuple is not found, this function returns with Bit 31 set (negative).
<code>_GET_NEXT_TUPLE</code>	<code>NPA_TUPLE_INFO_t</code> Filled structure describes the previous tuple retrieved. The tuple is filled with the next tuple data on return.	Retrieves the next tuple while enumerating tuple values. When the tuple sequence is exhausted, a zero returns.
<code>_WRITE_TLV_TUPLE</code>	<code>NPA_TUPLE_INFO_t</code> Filled structure describes the tuple to write.	Creates a new tuple in EEPROM or replaces the tuple if one already exists.
<code>_STORE_POST_ERROR</code>	<code>POST_ERROR_e</code> Sets the POST code to store.	Stores an 8-bit POST error code in EEPROM.
<code>_GET_POST_ERROR</code>	<code>POST_ERROR_e</code> Retrieves all 16 post error codes.	Retrieves all 16 8-bit POST error codes from EEPROM.
<code>_CLEAR_POST_ERROR</code>	<code>void</code> Clears all 16 POST error codes.	Clears or deletes all POST results from EEPROM.
<code>_TWSI_WRITE</code>	<code>TWSI_OP_t</code> Filled structure describes the following: <i>dev_addr</i> —I2C device address <i>addr</i> —Memory location <i>data</i> —Write data	Write a value to the specified TWSI device.
<code>_TWSI_READ</code>	<code>TWSI_OP_t</code> Filled structure describes: <i>dev_addr</i> —I2C device address <i>addr</i> —Memory location	Reads a value from the specified TWSI device.
<code>_PHY_WRITE</code>	<code>PHY_OP_t</code> Filled structure describes: <i>IdxNum</i> —PHY interface number <i>Reg</i> —Register number <i>Mask</i> —Bit mask to assert during write operation <i>Val</i> —Value to be written	Writes a value to a PHY register.
<code>_PHY_READ</code>	<code>PHY_OP_t</code> Filled structure describes: <i>IdxNum</i> —PHY interface number <i>Reg</i> —Register number <i>Mask</i> —Bit mask to assert during write operation <i>Val</i> —Returns the value read	Reads a value from the PHY register.
<code>_PHY_VS_WRITE</code>	<code>PHY_OP_t</code> Filled structure describes: <i>IdxNum</i> —PHY interface number <i>Reg</i> —Register number <i>Mask</i> —Bit mask to assert during write operation <i>Val</i> —Value to be written	Writes a value to the vendor specific region of the PHY space, typically, the SFP memory space.
<code>_PHY_VS_READ</code>	<code>PHY_OP_t</code> Filled structure describes: <i>IdxNum</i> —PHY interface number <i>Reg</i> —Register number <i>Mask</i> —Bit mask to assert during write operation <i>Val</i> —Returns the value read	Reads a value from the vendor specific regions of the PHY space, typically, the SFP memory space.

Command	Structure	Description
<code>_SET_LED_STATE</code>	Integer <code>LED_TYPE_e</code> flags Indicate LED state to set	Sets the LED state.
<code>_GET_LED_STATE</code>	Filled with <code>LED_TYPE_e</code> Flags indicating snapshot state on return.	Retrieves the current LED state.
<code>_DEVICE_COUNT</code>	Returns the number of boards in the system.	In PCI Target configuration, returns the number of PCI devices found.
<code>_PEEK</code>	<code>NPA_REG_RW_OP_t</code> <i>DevID</i> — Device IDs	In PCI Target configuration, peeks at an address.
<code>_POKE</code>	<code>NPA_REG_RW_OP_t</code> <i>DevIDx</i> —Device IDs	In PCI Target configuration, pokes at an address.
<code>_GET_BOARD_INFO</code>	<code>NPA_BOARD_INFO_t</code> structure Returns information about the FPGA, IPMC and MMC.	Retrieves version information for the board.
<code>_SET_ENVIRONMENT</code>	<code>NPA_ENV_ENTRY_t</code> structure Returns information about the environment variable specified in <i>envName</i> .	Initiates a 'setenv' for U-Boot environment variable.
<code>_GET_ENVIRONMENT</code>	<code>NPA_ENV_ENTRY_t</code> structure Sets the environment variable specified in <i>envName</i> to <i>envValue</i> .	Initiates a 'getenv' for U-Boot environment variable.
<code>_GET_NEXT_ENVIRONMENT</code>	<code>NPA_ENV_ENTRY_t</code> structure Returns environment information for the next environment variable after the one specified in <i>envName</i> .	Iterates through the environment list.
<code>_SET_PORT_ADMIN_STATE</code>	<code>NPA_PORT_INFO_t</code> structure Sets the port specified by the port on the device specified by <i>devId</i> to the mode specified by <i>eAdminState</i> .	Sets a port state to 'up' or 'down'; Allows 'up' on link or forces 'down' on link.
<code>_GET_PORT_ADMIN_STATE</code>	<code>NPA_PORT_INFO_t</code> structure Returns the administrative state on the port specified by the port on the device specified by <i>devId</i> .	Queries the current admin 'up' or 'down' state.
<code>_GET_PORT_STAT</code>	<code>NPA_PORT_INFO_t</code> structure Return the statistic specified by <i>statIndex</i> . The name of the statistic at <i>statIndex</i> returns in <i>statName</i> , and the counter value returns in <i>statHigh</i> and <i>statLow</i> .	Queries and retrieves information about the port state.
<code>_SET_PORT_STATE</code>	<code>NPA_PORT_INFO_t</code> structure Sets the port specified by the port on the device specified by <i>devId</i> to the operational mode specified by <i>eOperState</i> .	Queries and sets information pertaining to the port state.
<code>_GET_PORT_STATE</code>	<code>NPA_PORT_INFO_t</code> structure Returns the operational mode of the port specified by the port on the device specified by <i>devId</i> .	Retrieves the statistic at the specified index number.
<code>_GET_PORT_STAT_COUNT</code>	<code>NPA_PORT_INFO_t</code> structure Returns the number of statistics available to the specified device.	Retrieves a count of the number of statistics that the port provides.
<code>_RESET_PORT_STATS</code>	<code>NPA_PORT_INFO_t</code> structure Resets the statistical counters for the device specified by <i>devId</i> .	Resets the statistic counters.
<code>_SET_LOOPBACK</code>	<code>NPA_LOOPBACK_INFO_t</code> structure Sets the port specified by the port on the device specified by <i>devId</i> to the loopback mode specified by <i>type</i> .	Sets the loopback state using <code>NPA_LOOPBACK_INFO_t</code> .
<code>_GET_LOOPBACK</code>	<code>NPA_LOOPBACK_INFO_t</code> structure Returns the loopback mode of the port specified by <i>port</i> on the device specified by <i>devId</i> .	Gets the loopback state via <code>NPA_LOOPBACK_INFO_t</code> .
<code>_GET_RTM_INFO</code>	<code>NPA_RTM_INFO_t</code> filled structure describes: <i>Board_type</i> —Set to the value of the board module type. <i>Mmc_info</i> —Contains all of the MMC revision information.	Returns information about the attached RTM, if present.

Command	Structure	Description
<code>_NPA_DEV_EX_CMD=0xF0</code>	<code>NPA_DEV_EX_CMD_t</code> structure Sends an uninterpreted IOCTL command to the device specified in <i>devId</i> .	Extended device specific commands = 0xF0.
<code>eNpaZ130_SetSynceConfig</code>	<code>NPA_SYNCE_CFG_t</code> Filled structure describes: <i>dpll_mode</i> —Specifies the desired clocking mode. <i>primary</i> —Specifies the primary clock reference lane source. <i>secondary</i> —Specifies the secondary clock reference lane source.	Configures the clock synchronization mode and sources for the WANic-66512.
<code>eNpaZ130_GetSynceConfig</code>	<code>NPA_SYNCE_CFG_t</code> Filled structure describes: <i>dpll_mode</i> —Specifies the current clocking mode. <i>primary</i> —Specifies the current primary clock reference lane source. <i>secondary</i> —Specifies the current secondary clock reference lane source.	Retrieves information about the clock synchronization mode and sources for the WANic-66512.
<code>eNpaZ130_GetSynceStatus</code>	<code>NPA_SYNCE_STATUS_t</code> Filled structure describes: <i>Holdover_status</i> —Describes the current holdover state <i>Lock_status</i> —Describes the currently selected clock reference quality. <i>Ref_clk</i> —Describes the currently used reference clock source.	Retrieves information about the clock synchronization status for the WANic-66512.

IOCTL Command Types are shown in Figure 4-7.

Figure 4-7 IOCTL Command Types

```
typedef struct _NPA_TUPLE_INFO {
    octeon_eeprom_header_t hdr;
    uint8_t data[ OCTEON_EEPROM_MAX_TUPLE_LENGTH ];
} NPA_TUPLE_INFO_t;

typedef struct _TWSI_OP {
    uint8_t device_id;    /**< TWSI device ID */
    uint16_t addr;       /**< address to perform operation on */
    uint16_t data;       /**< data to write or data that was read */
    uint16_t mask;       /**< mask to apply while writing data to address */
} TWSI_OP_t;

typedef struct _PHY_OP {
    uint8_t     idxNum;   /**< PHY device ID */
    uint16_t    reg;      /**< Address to perform operation on */
    uint16_t    mask;     /**< Mask to apply while writing data to address */
    uint16_t    val;      /**< Data to write or data that was read */
} PHY_OP_t;

typedef struct
{
    uint8_t running_rev_major;    /**< The IPMC currently running revision */
    uint8_t running_rev_minor;    /**< The IPMC currently running revision */
    uint8_t active_rev_major;     /**< The active IPMC image revision */
    uint8_t active_rev_minor;     /**< The active IPMC image revision */
    uint8_t backup_rev_major;     /**< The backup IPMC image revision */
    uint8_t backup_rev_minor;     /**< The backup IPMC image revision */
    uint8_t failsafe_rev_major;   /**< The failsafe IPMC image revision */
    uint8_t failsafe_rev_minor;   /**< The failsafe IPMC image revision */
    uint8_t soak_counter;        /**< The soak test counter number */
    uint8_t active_image_number;  /**< The active image number */
    uint8_t hardware_rev;        /**< The IPMC hardware revision number */
    uint8_t test_mode;           /**< The IPMC test mode */
} NPA_IPMC_INFO_t;

typedef NPA_IPMC_INFO_t NPA_MMC_INFO_t;

typedef struct _NPA_BOARD_INFO {
    uint32_t driver_rev;          /**< Version number for the active BSP
                                   version */
    uint32_t boot_normal_rev;     /**< Version number for the UBOOT Normal
                                   Partition if available */
    int32_t  boot_failsafe_rev;   /**< Version number for the UBOOT Failsafe
                                   Partition if available */
    uint32_t fpga_rev;           /**< Version number for the FPGA */
    uint8_t  bPciHostMode;       /**< Set to TRUE(1) if the BSP is running in
                                   PCI HOST Mode, FALSE(0) in PCI Target Mode*/
    uint16_t board_type;         /**< Set to the value of the Board Module
                                   Type */
    uint8_t  boot_normal_active;  /**< Indicates if the Normal or Failsafe
                                   Partition is active */
    uint8_t  boot_activate_flash; /**< Indicates if the Primary or Backup
                                   Flash is active */
    uint8_t  npu_id;             /**< Letter ID of the current Octeon NPU*/
    uint16_t mcr;                /**< Master control register value */
    NPA_IPMC_INFO_t ipmc_info;   /**< Contains all of the IPMC revision
                                   information and more */
    NPA_MMC_INFO_t mmc_info;     /**< Contains all of the MMC revision
                                   information and more */
} NPA_BOARD_INFO_t;
```

Figure 4-7 (continued) IOCTL Command Types

```

typedef struct NPA_RTM_INFO_t {
    uint32_t board_type      /**< Set to the value of the Board Module Type */
    NPA_MMC_INFO_t mmc_info; /**< Contains all the MMC revision info and more */
} NPA_RTM_INFO_t;

typedef struct _NPA_SYNC_CFG_t {
    unsigned int dpll_mode; /**< Specifies syncE mode:
                                0 = Freerun
                                1 = Forced holdover
                                2 = Reference lock
                                3 = Automatic */
    unsigned int primary;   /**< Primary reference clk port 0-3 */
    unsigned int secondary; /**< Secondary reference clk port 0-3 */
} NPA_SYNC_CFG_t;

typedef struct _NPA_SYNC_STATUS_t {
    unsigned int holdover_status; /**< Holdover mode status,
                                    1 = Holdover, 0 = Normal */
    unsigned int lock_status;     /**< Reference lock status, 1 = ok */
    unsigned int ref_clk;        /**< If running in auto mode, current ref clk */
} NPA_SYNC_STATUS_t;

typedef struct _NPA_REG_RW_OP {
    uint32_t devId; /**< Specifies the device to perform the operation on */
    uint64_t addr;  /**< Specifies the address of the register */
    uint64_t data;  /**< Specifies the data qword to write or the data qword read */
    uint64_t mask;  /**< Specifies the data mask used during register write */
} NPA_REG_RW_OP_t;

typedef struct _NPA_LOOPBACK_INFO_t {
    eNpaFunction devId; /**< Specifies the device id to use for operation */
    uint16_t port;      /**< If a port is necessary, specifies port number */
    eNpaLoopbackTypes type; /**< Specifies the type of loopback */
} NPA_LOOPBACK_INFO_t;

typedef struct _NPA_PORT_INFO_t {
    eNpaFunction devId; /**< Specifies the device id to use for operation */
    uint16_t port;      /**< If a port is necessary, specifies port
                                number */
    eNpaOperState eOperState; /**< Sets or gets the port operational state */
    eNpaAdminStae eAdminState; /**< Sets or gets the port admin state */
    uint32_t statIndex; /**< The count of statistics, or the statistic
                                to retrieve */
    char statName[128]; /**< The statistic friendly name */
    int32_t statHigh; /**< The high 32 bits of the statistic */
    uint32_t statLow; /**< The low 32 bits of the statistic */
} NPA_PORT_INFO_t;

typedef struct _NPA_ENV_ENTRY_t {
    char envName[NPA_MAX_ENVNAME_SIZE];
    char envValue[NPA_MAX_ENVVALUE_SIZE];
} NPA_ENV_ENTRY_t;

typedef struct _NPA_DEV_EX_CMD_HEADER_t {
    NpaFunction devId; /**< Specifies the device id to use for the
                                operation */
    int cmd;
    int cmdLen;
} NPA_DEV_EX_CMD_HEADER_t;
#define NPA_DEV_EX_CMD_HEADER_SIZE sizeof(NPA_DEV_EX_CMD_HEADER_t)

```

Figure 4-7 (continued) IOCTL Command Types

```
typedef struct _NPA_DEV_EX_CMD_t {
    eNpaFunction devId;          /**< Specifies the device id to use for the
                                operation */
    int          cmd;
    int          cmdLen;
    uint8_t     cmdBuf[(500 - sizeof(unsigned int)) - NPA_DEV_EX_CMD_HEADER_SIZE];
} NPA_DEV_EX_CMD_t;

#define NPA_DEV_EX_CMD_SIZE sizeof(NPA_DEV_EX_CMD_t)

typedef struct _NPA_IOCTL_CMD {
    unsigned int devNumber;
    union {
        NPA_TUPLE_INFO_t      TupleInfo;
        TWSI_OP_t             TwsiInfo;
        PHY_OP_t              PhyopInfo;
        NPA_BOARD_INFO_t      BoardInfo;
        NPA_LOOPBACK_INFO_t   LoopbackInfo;
        NPA_PORT_INFO_t       PortInfo;
        NPA_ENV_ENTRY_t        EnvInfo;
        NPA_REG_RW_OP_t        RegInfo;
        NPA_DEV_EX_CMD_t       DeviceExtendedInfo;
        unsigned char          error_code[ 16 ];
        unsigned char          ledState;
        unsigned char          buffer[500 - sizeof(unsigned int)];
    };
    /* IOCTL structure is always 512 bytes */
} NPA_IOCTL_CMD_t;
```

4.5.4.3 Exported Kernel Symbols

The NPA Driver also contains Exported Kernel Symbols, which are global kernel functions. Figure 4-8 identifies the kernel functions that are exported from the NPA Driver.

Figure 4-8 Exported Kernel Symbols.

```
int npaHwWdog_SetPreTimeout( int pretimeo );
int npaHwWdog_GetPreTimeout( void );
int npaHwWdog_SetTimeout( int timeo );
int npaHwWdog_GetTimeout( void );
int npaHwWdog_Pet( void );
int npaHwWdog_EnableWdog( int pretimeo, int timeo );
int npaWdog_SetNotifier( void (*wdog_event_handler)( void *, int cause ),
void * param );

#if defined(NPA_CONFIG_FAULT)
    int npa_bind_fault_reporter ( void (*fault_reporter) ( int size, void *
data_ptr ));
    int npa_unbind_fault_reporter ( void );
    int npa_set_fault_processing( eNpaFunction eFunctionID, int bEnable );
    int npa_bind_event_reporter ( void (*event_reporter) ( int size, void *
data_ptr ));
    int npa_unbind_event_reporter ( void );
#endif

struct npaObject_t; /* unused when called externally in PCI_HOST_MODE */
u64 npaGetBootCfgAddr( struct npaObject_t * devObj, unsigned int chipssel
);
```

4.5.5 Linux In-Service Daemon

The In-Service Daemon provides a runtime background process that continuously checks for device responsiveness, monitoring hardware and alerting the system if a device becomes un-responsive. The In-Service Daemon is dependent upon the NPA Driver.

The In-Service Daemon logs all errors to `/var/log/syslog`, and runs in both foreground and daemonized mode (background). For more information on the In-Service Daemon, see [Chapter 5: Linux Support Package \(LSP\) Applications](#).

4.5.6 Diagnostics Application

The diagnostics application allows you to perform data loop tests to exercise all or select Ethernet ports with configurable data options. The application gives EEPROM access for setting LABI, SIPI, PFI, SFI, Auto Scripting, MAC, and board information EEPROM configurations. All POST and data loopback test results can also be retrieved from the EEPROM via the Linux Diagnostic Application. The Diagnostic Application also provides an upgrade utility for U-Boot and Flash application image. The diagnostics application also provide peek/poke functionality for all available devices on the board. For more information on the Diagnostic Application, see [Chapter 5: Linux Support Package \(LSP\) Applications](#).

4.6 Additional Features

The LSP provides the following additional features:

- Interrupt generation
- Exception processing
- In-Service testing
- Memory mapping
- Testing POST results

Interrupt Generation

The LSP provides hardware configuration for interrupt generation caused by hardware failures that do not result in errors through exception processing. The introduces all interrupt request (IRQ) definitions and Linux IRQ handling infrastructures changes required to permit the binding of Linux IRQ handlers into the kernel through the standard `request_irq` interface.

Exception Passing

The LSP provides processor exception handling to the point of full trace back dumps to the standard Linux log.

In-Service Testing

The LSP provides in-service tests for each major device. It provides a test plan, which lists executed test cases with test results to verify complete functionality and known errata list. It also has an automated test suite that verifies functionality.

Memory Mapping

The LSP provides a memory map, which identifies all reserved areas that are not available to Linux. It identifies the differences between physical memory and that which is reported as memory.

POST Results

The LSP also supports retrieval of POST results from non-volatile memory.

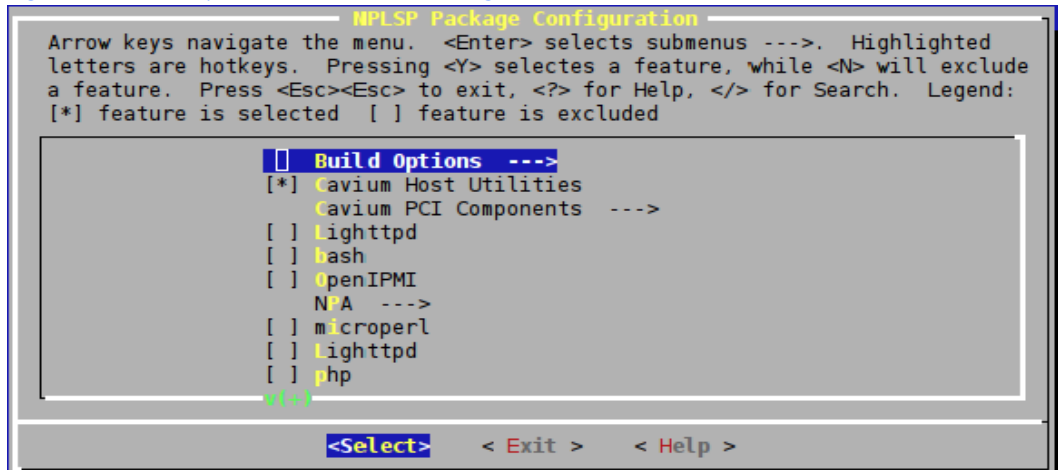
4.7 Examples

For your convenience, this section describes Linux and Simple Executive application examples, which are also available on the software CD-ROM in the following directories:

- `$OCTEON_ROOT/cav-gefes/examples/linux`
- `$OCTEON_ROOT/cav-gefes/examples/simple_exec`

Examples in this section refer to the NPLSP Configuration Menu *similar* to that shown in Figure 4-9. The NPLSP Configuration Menu is part of the LSP Package, which contains all the components to support software development for the OCTEON Multi-Core Processor or the Embedded Debian Linux environment on a WANic-66512 packet processor.

Figure 4-9 Example of the NPLSP Configuration Menu



4.7.1 Linux Applications Examples

The `$OCTEON_ROOT/cav-gefes/examples/linux` directories contains folders for the following two Linux application examples:

- Hello
- Board Information (boardinfo)

Example 1: Hello

This is a basic 'Hello World' Linux application. To build this example, enable the 'Examples' option in the NPLSP Configuration Menu, or run the 'make all' command from the ``$OCTEON_ROOT/cav-gefes/examples/linux/hello'` directory.

When successfully compiled, the binary 'Hello' is built in the current directory, as well as copied into the 'Extra-files' directory of the embedded file system.

After rebuilding the embedded kernel, run the command ``/bin/hello'` command. Output displays as follows:

```
~ # /bin/hello
Hello world
```

Example 2: Board Information

This example performs simple Open, IOCTL, and Close calls to the NPA Driver, and displays current information about the board.

To build this example, enable the 'Examples' option in the NPLSP Configuration Menu, or run the 'make all' command from the '\$OCTEON_ROOT/cav-gefes/examples/linux/boardinfo' directory.

When successfully compiled, the binary 'boardinfo' is built in the current directory, as well as copied into the 'Extra-files' directory of the embedded file system.

After rebuilding the embedded kernel, run the command:

```
# /bin/boardinfo
```

Output displays similar to the following:

```
~ # /bin/boardinfo
Running Version           : x.xx.x
Running UBoot Failsafe   : x.x.x
Running UBoot Normal     : x.x.x
Running FPGA Version     : x.x
Running in PCI mode      : host
Running on NPU           : A
```

4.7.2 Simple Exec Application Examples

The '\$OCTEON_ROOT/cav-gefes/examples/simple_exec' directories contains folders for two Simple Executive application examples:

- Hello
- Intercept

Example 3: Hello

This example application performs a simple loop on all active cores displaying Hello World with an incrementing counter.

To build this example, enable the 'Examples' option in the NPLSP Configuration Menu, or run the 'make all' command from the '\$OCTEON_ROOT/cav-gefes/examples/simple_exec/hello' directory.

When successfully compiled, the binary 'hello_exe' is built in the current directory. To run Simple Executive applications, perform the following steps:

1. Copy the 'hello_exe' Simple Executive application to a TFTP server.
2. Set a static IP address, or obtain an IP address through DHCP from U-Boot:

```
# dhcp
```

or

```
# set ipaddr <i.p. address>
```

3. Set the TFTP server IP address in U-Boot, for example:

```
# set serverip 192.168.1.1
```

4. From U-Boot, transfer the simple exec binary to the target system:

```
# tftpboot 20000000 hello_exe
```

5. Boot the simple exec application:

```
# bootoct 20000000 coremask=3ff
```

Output similar to the following displays approximately once every second:

```
PP0:~CONSOLE-> Core[0]: Hello world - 0
PP2:~CONSOLE-> Core[2]: Hello world - 0
PP4:~CONSOLE-> Core[4]: Hello world - 0
PP7:~CONSOLE-> Core[7]: Hello world - 0
PP9:~CONSOLE-> Core[9]: Hello world - 0
PP10:~CONSOLE-> Core[10]: Hello world - 0 PP6:~CONSOLE-
> Core[6]: Hello world - 0 PP5:~CONSOLE-> Core[5]:
Hello world - 0 PP8:~CONSOLE-> Core[8]: Hello world - 0
PP11:~CONSOLE-> Core[11]: Hello world - 0 PP1:~CONSOLE-
> Core[1]: Hello world - 0 PP3:~CONSOLE-> Core[3]:
Hello world - 0
```

Example 5: Intercept

This application example intercepts incoming packets to any of the Cavium Ethernet ports, and drops the packets if the size exceeds a certain limit.

The application starts up and waits for Linux to load the Cavium Ethernet driver. This application runs in conjunction with a booted Linux kernel and the Cavium Ethernet driver, and does not work properly unless the Cavium Ethernet driver is loaded.

When the driver is loaded, all the Cavium Ethernet ports are configured so that all incoming packets are received by the application instead of Linux.

Then, all packets display on the screen, and any packets exceeding the maximum size limit are dropped. Packets within the maximum size limit are forwarded on to Linux.

To build this example, enable the 'Examples' option in the WANic-66512 NPLSP Configuration Menu, or run the 'make all' command from the '\$OCTEON_ROOT/cav-gefes/examples/simple_exec/intercept' directory.

When successfully compiled, the binary 'intercept_exe' is built in the current directory. To run the Simple Executive applications, perform the following steps:

1. Copy the 'intercept_exe' Simple Executive application to a TFTP server.
2. Copy the embedded kernel image to a TFTP server.
3. Set a static IP address, or obtain an IP address through DHCP from U-Boot:

```
# dhcp
```

4. Set the TFTP server ip address in U-Boot. For example, enter:

```
# set serverip 192.168.1.1
```

5. From U-Boot, transfer the simple exec binary to the target system:

```
# tftpboot 30000000 intercept_exe
```



NOTE

Ignore messages that indicate that data is being loaded outside of the reserved load area.

6. From U-Boot, transfer the Linux kernel to the target system:

```
# tftpboot 20000000 vmlinux-1.26.1
```

7. Boot the Linux kernel on the upper cores:

```
# bootoctlinux 20000000 coremask=30
```

8. Boot the Simple Executive application on the lower cores:

```
# bootoct 30000000 coremask=00f
```

Once booted, the serial console displays both Linux and Simple Executive output and also accepts Linux input. The default IP addresses for the four Cavium Ethernet ports are:

```
xauio - 192.168.0.100
```

```
xauil - 192.168.1.100
```

9. Change at least one of the IP addresses to a valid address on your subnet by running the following Linux command:

```
# ifconfig <interface> <i.p. address>
```

For example, # ifconfig eth0 192.168.0.100

10. To test the application, ping one of the Ethernet interfaces from another system on your subnet:

```
# ping <i.p. address>
```

When the ping is successful, Simple Executive output displays similar to the following:



NOTE

The Simple Executive application handles the packet, and then forwards it to Linux.

```
PP0:~CONSOLE-> Packet work group: 0
PP0:~CONSOLE-> Packet Length: 60
PP0:~CONSOLE-> Input Port: 16
PP0:~CONSOLE-> QoS: 0
PP0:~CONSOLE-> Buffers: 1
PP0:~CONSOLE-> Buffer Start:41d6cc080
PP0:~CONSOLE-> Buffer I : 0
PP0:~CONSOLE-> Buffer Back: 1
PP0:~CONSOLE-> Buffer Pool: 0
PP0:~CONSOLE-> Buffer Data: 41d6cc140
PP0:~CONSOLE-> Buffer Size: 1856
PP0:~CONSOLE-> 0180c20000000002
PP0:~CONSOLE-> 7ef02e4c00264242
PP0:~CONSOLE-> 0300000000008000
PP0:~CONSOLE-> 000142efafca0000
PP0:~CONSOLE-> 073f800000d001cd
PP0:~CONSOLE-> 206380ed02001400
PP0:~CONSOLE-> 02000f0000000000
PP0:~CONSOLE-> 00000000
```

11. Next, ping one of the Ethernet interfaces while specifying a size *bigger* than the maximum packet size limit. The maximum packet size is 1024, which can be changed in the example by changing the size of the defined 'MAX_PACKET_SIZE' inside of 'intercept.c' and then rebuilding.

```
# ping -s 1024 <i.p. address>
```

A message similar to the following displays to indicate that packets are too large and are being dropped:

```
PP0:~CONSOLE-> Rcvd 1066 byte pkt that exceeds max size of
1024, dropping being displayed on the simple exec console.
The packet is being dropped without ever being forwarded to
Linux.
```


5 • Linux Support Package (LSP) Applications

This chapter describes the WANic-66512 Linux Support Package, which includes the Diagnostic Utility and the In-Service Daemon. This chapter describes how to run tests to verify and to configure the functionality of components on the WANic-66512, as well as a process to continuously monitor the hardware.

5.1 Diagnostic Utility

The Diagnostic Utility allows the user to perform data loop tests, which can exercise all or selected Ethernet ports with configurable options. The Diagnostic Utility gives EEPROM access for setting U-Boot parameters, such as LABI, SIPI, PFI, SFI, MAC, and board information EEPROM configurations. All POST and data loopback test results can also be retrieved from the EEPROM through the Diagnostic Utility. The Diagnostic Utility also provides an upgrade utility for U-Boot and Flash application images.

The WANic-66512 Diagnostic Utility is a Linux image that provides the following:

- Transmit/Receive Tests – verify the WANic-66512 Ethernet interfaces.
- Configuration Options – configure the EEPROM to specify boot parameters.
- PCIe Host Utilities – provide services for booting U-Boot over the PCIe interface
- POST and Test Results – verify the most recent POST diagnostics and automatic transmit/receive results.

The Diagnostic Utility also provides a log file and a status file. The log file contains various diagnostic utility information. The status file stores data-loop test results.



NOTE

Features and screens captures displayed in this manual may vary from those displayed by your software. Please consult with a GE Intelligent Platforms Customer Technical Support Engineer to make sure you install the latest software update.

Command Line Parameters

In addition, the Diagnostic Utility provides command line parameters to run tests, and to set or display select diagnostic parameters. Table 5-1 lists the Command Line Parameters. When you enter the **-h** (elp) parameter, a screen similar to Figure 5-1 displays all the Command Line Parameters and settings (where applicable.)

Table 5-1 Command Line Parameters

Test Parameters and Settings	Description
--resultshow=[all/post/dataloop/memtests]	Default is to show all.
--display=yes	Displays the results and exits.
--showtp=yes	Displays throughput during data-loop test.
--skipmemtest=yes	Skips auto-run memory tests, use with -a option.
--skipdatatest=yes	Skips auto-run data-loop test, use with -a option.
--skipusbtest=yes	Skips USB storage test, use with -a option.
-a	Automatically runs the data loop and memory tests from the command line.
-f	Enables FCC mode.
-h	Displays all the command line parameters.
-l	Sets the name for the diagnostic Log file.
-m	Sets the path to diagnostics Log and Status files.
-o	Puts the interface ports in loop mode.
-r	Sets the name for the diagnostic Status file.
-s	Sets the packet frame size.
-t	Sets the data loop test time duration.

Figure 5-1 Command Line Parameters

```
# npaDiag -h

--resultshow=[all/post/dataloop/memtests]  default is show all
--display=yes                               displays the results and exits
--showtp=yes                               display throughput during data-loop test
--skipmemtest=yes                          skips auto-run memory tests, use with -a option
--skipdatatest=yes                         skips auto-run data-loop test, use with -a option
--skipusbtest=yes                          skips USB storage test, use with -a option

-l <logfile>: set the logfile location
-r <status>: set the status file location
-s <size>: set the frame size
-t <time>: set the test time in seconds
-m <log dir>: set the log file directory location
-f run FCC dataloop test mode
-a auto run dataloop test
-o put interface ports in loopmode
```


5.1.1 Setup

To run the Diagnostic Utility, you must have the following:

- WANic-66512 module installed according to the directions provided by in “*Chapter 1: Getting Started.*”
- Host serial or network terminal.
- Ethernet connection from host terminal to the WANic-66512, or serial connection to the UART.

Connect to the WANic-66512, using either Telnet or a serial connection to the host terminal with a serial cable.

- When using Telnet with the factory-installed Linux application, log into a Telnet session to connect to the Linux target running on the WANic-66512. Enter the following Telnet command:

```
telnet 192.168.x.100
```

where *x* is the Ethernet Interface Number 0–1

- When using a serial connection, connect one end of a serial cable to the 5-pin connector on the WANic-66512 SDA and the other end to the host terminal. Use a terminal setting of 115200n1.

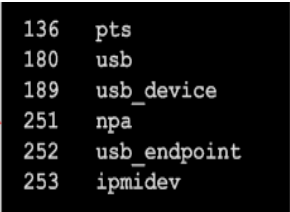
NPA Character Device

Create the NPA character device, which allows the user application to pass commands to the NPA Driver. To create the NPA character device, perform the following steps after the `npaDriver.ko` module is installed.

1. Get the OCTEON device major number by entering the following command:
cat /proc/devices

A listing similar to that shown in Figure 5-2 displays. The major number is the number associated with the `npa`.

Figure 5-2 OCTEON NPA Major Number



```
136 pts
180 usb
189 usb_device
251 npa
252 usb_endpoint
253 ipmidev
```

Major number →

2. Create the device by entering the following command and inserting the major number.

```
# mkmmod /dev/npa c<major-number>0
```

5.1.2 Running the Diagnostic Utility

To run the Diagnostic Utility, perform the following Steps:

1. Start the Diagnostic Utility.

Enter the following command to start the Diagnostic Utility:

```
# npaDiag <option>
```

When the Main Menu displays, it prompts you to enter a Menu Choice.

Figure 5-3 displays the PCI Host Mode screen. Table 5-2 describes PCI Host Mode options on the Diagnostic Utility Main Menu.



NOTE

The vmlinux image kernel/file system included in the LSP package supports the Diagnostic Utility. When the vmlinux image is executed, the Linux application is configured to conduct an automated loop test following the Linux boot process and exercises all Ethernet ports.

Figure 5-3 Diagnostic Utility Main Menu

```
Running Octeon Model           : CM66xx
Running Driver Version         : x.xx.x
Running UBoot Failsafe        : x.x.x
Running UBoot Normal          : x.x.x
Running FPGA Version          : x.x
Running in PCI mode           : host
Running on NPU                 : x
Setting cpuindex 0 for eth0
Setting cpuindex 1 for eth1
Setting cpuIndex x for xau10
NPA_mem_test namealloc is not set, skip DRAM memory test
***          WANic-66xx Diagnostics vx.xx.x NPU: X
1. View Test Results          10. Flash Menu
2. EEPROM Menu                11. Log File locations
3. Reserved                   12. Reserved
4. Peek/Poke menu             13. MEM Test
5. Run Auto Test              14. Reserved
6. Auto Test Configuration Menu 15. USB Storage Test
8. Diagnostic LED test
0. Exit
Menu Choice (0) : █
```

Table 5-2 Diagnostic Utility Main Menu

Menu Item	Description
1. View Test Results	Displays the results from all the diagnostic tests performed automatically during the most recent boot of the board.
2. EEPROM Menu	Displays the EEPROM Menu.
3. Reserved	
4. Peek/Poke Menu	Allows the peeking and poking of board components.
5. Run Auto Test	Executes the transmit/receive test using the settings in Table 5-4.
6. Auto Test Configuration Menu	Displays the Auto Test Configuration Menu.
8. Diagnostic LED Test	Toggles the LED on and off.
10. Flash Menu	Displays options to view and update Flash partitioning.
11. Log File Location	Display the path to the log files.
12. Reserved	
13. Mem Test	Conducts memory read/write tests for PMEM and/or SDRAM. Only SDRAM test is allowed to run when <code>npa_mem_test</code> named <code>allocated space</code> is set in U-Boot prior to booting Linux. (For example, named <code>alloc npa mem test <size> <addr>.</code>)
14. Reserved	
15 USB Storage Test	Conducts USB storage device read/write test if device is available and mounted.
0. Exit	Leaves the Diagnostic Utility.



NOTE

Where appropriate, default values are shown in angle brackets <>.

5.2 Main Menu Options

A description of each Main Menu option follows.

View Test Results

Select **1. View Test Results** to display the most recent POST test results and the most recent Auto Test results. (See option **5. Run Auto Test.**) The Diagnostic Utility displays PASSED or FAILED results from the WANic-66512 most recent boot, similar to that shown in Figure 5-4.

Figure 5-4 View Test Results Display

```
Menu choice (0) : 1
POST status
  POST Passed
Diagnostic Status
Retrieving data-loop results file
  eth0: PASSED
  eth1: PASSED
Retrieving DDR3 memory test result file....
  cpu0: All Memory Tests: PASSED
  cpu1: All Memory Tests: PASSED
  cpu2: All Memory Tests: PASSED
  cpu3: All Memory Tests: PASSED
  cpu4: All Memory Tests: PASSED
  cpu5: All Memory Tests: PASSED
  cpu6: All Memory Tests: PASSED
  cpu7: All Memory Tests: PASSED
  cpu8: All Memory Tests: PASSED
  cpu9: All Memory Tests: PASSED
Retrieving LED test results ....
  LED test: PASSED
Hit enter to continue .....
```

EEPROM Menu

Select **2. EEPROM Menu** to display the EEPROM Menu options as shown in Figure 5-5. Use these options to modify the EEPROM contents on the WANic-66512.

Figure 5-5 EEPROM Menu

```
Menu Choice (0) : 2
1. Write MAC address
2. Write Board Description
3 Set Source IP Information
4. Set Primary File Information
5. Set Secondary File Information
6. Set Load and boot Information
7. Display SIPI/PFI/SFI/LABI Information
8. Clear POST codes
Menu Choice (0) :
```

Table 5-3 EEPROM Menu Options

Option	Description
1. Write MAC Address	Displays the MAC address for the WANic-66512 and also allows the programming of a new MAC address.
2. Write Board Description	Sets the revision and serial numbers for the module.
3. Set Source IP Information	Sets the source IP address information (SIPI) for each Ethernet interface.
4. Set Primary File Information	Defines the Primary File Information (PFI) definitions.
5. Set Secondary File Information	Defines the Secondary File Information (SFI) definitions.
6. Set Load and Boot Information	Defines the Load and Boot Information (LABI) definitions.
7. Display SIPI/PFI/SFI/LABI Information	Shows all the SIPI, PFI, SFI and LABI information.
8. Clear POST codes	Erases all POST codes.
0. Exit	Leaves the EEPROM Menu, initializes the EEPROM, and returns to the Main Menu.

Run Auto Test

Select **5. Run Auto Test** to run a transmit/receive test using the parameters specified in option **6. Auto Test Configuration Menu**. When the Auto Test completes, the latest test result is updated.



NOTE

If a Telnet connection is established to the WANic-66512, the connection may be lost when the Auto Test executes.

Figure 5-6 Run Auto Test

```
Iter6:
Inter-Octeon Tx/Rx Threads:
spi0:3497,3334 spi1:3497,3340 spi2:3496,3336 spi3:3495,3336 spi4:3494,3330 spi5:3495,3342 sp
16:3494,3344 spi7:3494,3341 spi8:3494,3339 spi9:3494,3339
RTM Tx/Rx Threads:
spi10:3494,3492 spi11:3504,3480 spi12:3503,3480 spi13:3492,3487 spi14:3492,3482 spi15:3492,34
75 spi16:3492,3477 spi17:3491,3470 spi18:3502,3489 spi19:3502,3484

Iter7:
Inter-Octeon Tx/Rx Threads:
spi0:3998,3833 spi1:3998,3841 spi2:3997,3835 spi3:3996,3835 spi4:3995,3830 spi5:3996,3842 sp
16:3995,3842 spi7:3995,3841 spi8:3995,3838 spi9:3995,3838
RTM Tx/Rx Threads:
spi10:3995,3992 spi11:4005,3979 spi12:4003,3978 spi13:3993,3985 spi14:3993,3980 spi15:3993,39
73 spi16:3992,3975 spi17:3992,3968 spi18:4003,3987 spi19:4003,3985

Iter8:
Inter-Octeon Tx/Rx Threads:
spi0:4496,4328 spi1:4496,4336 spi2:4495,4331 spi3:4494,4330 spi4:4493,4323 spi5:4494,4338 sp
16:4493,4340 spi7:4493,4338 spi8:4493,4335 spi9:4493,4331
RTM Tx/Rx Threads:
spi10:4493,4492 spi11:4505,4475 spi12:4504,4473 spi13:4491,4484 spi14:4491,4476 spi15:4491,44
66 spi16:4491,4470 spi17:4490,4466 spi18:4503,4484 spi19:4503,4482
```

Auto Test Configuration Menu

Select **6. Auto Test Configuration Menu** to display test configuration options as shown in Figure 5-8. Use these menu options to modify the configuration before selecting **5 Run Auto Test**.

CAUTION

A power cycle of the WANic-66512 causes the configuration to be lost.

To test the WANic-66512, attach a Category 5 Enhanced (Cat 5E) cable to each of the ports as shown in Figure 5-7. Connect Port 0 to Port 1.

Figure 5-7 Test Configuration with Cable Attachment

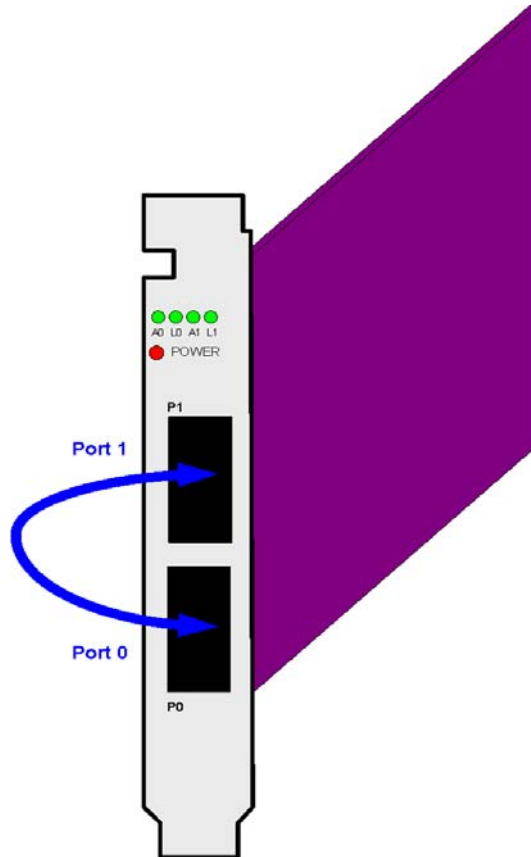


Figure 5-8 Auto Test Configuration Menu

```
Menu Choice (0) : 6
1. Frame Size
2. Enable/Disable Port to be tested
3. Number of seconds to run test
4. Reserved
5. Put Interfaces into Line Loop Mode [DISABLED]
6. Stop loop test on failure [DISABLED]
8. Display data-loop throughput [DISABLED]
9. Put Interfaces into Self Peer Mode [DISABLED]
0. Exit
Menu Choice (0) :
```

Table 5-4 Test Configuration Menu Options

Option	Description	Default
1. Frame Size	Specifies the frame size of the transmitted packets [96 – 1500].	1500
2. Enable/Disable Port to be Tested	Toggles to enable/disable testing on the specified port.	Enable All
3. Number of Seconds to run test	Specifies the number of seconds to run the test. [1 – 86401] (24 hours+1 second). -1 = continuous	30 seconds
4. Reserved		
5. Put interfaces into Line Loop Mode [DISABLED]	Loops incoming data back out the same interface.	Disabled
6. Stop loop test on failure [DISABLED]	When data-loop test exceeds a specified error threshold limit, the data loop test stops.	Disabled
7. Enter EPS value	Sets stop on error threshold value. NOTE: This option is set only when Option 6 is enabled.	
8. Display data-loop throughput [DISABLED]	Shows transmit and receive data throughput.	Disabled
9. Put Interfaces into Self-Peer Mode [DISABLED]	Interface sends data out and expects to receive data on the same interface. The Port is paired with itself.	Disabled
0. Exit	Leaves the Test Configuration Menu and returns to the Main Menu.	None

Diagnostic LED Test

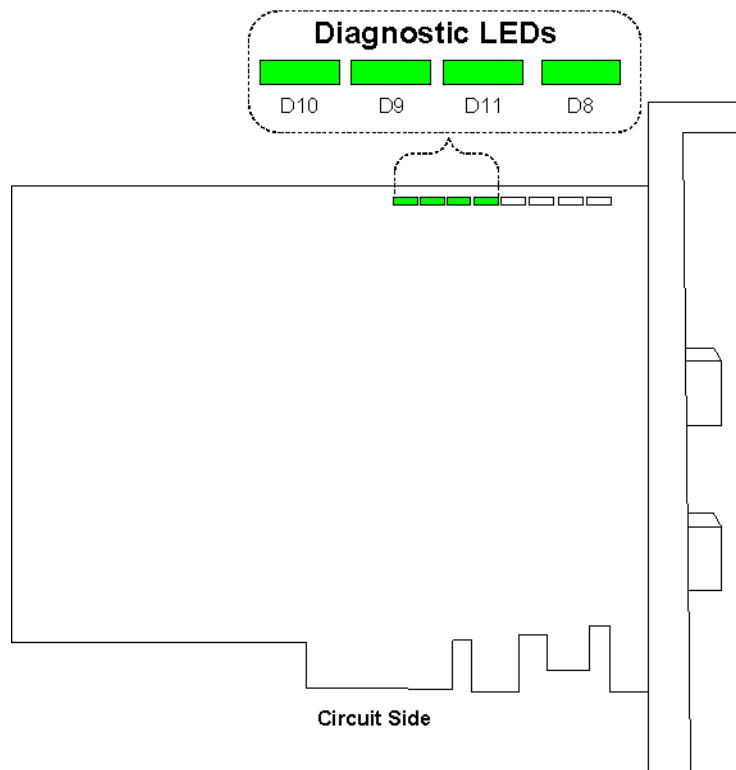
Select **8. Diagnostic LED Test** to verify that the test is toggling the LEDs on and off. This option prompts you with a message as shown in Table 5-5. View the Diagnostic LEDs, located approximately as shown in Figure 5-9, to verify that the LEDs are turning on and off.

To stop the test, press the **<Enter>** key.

Table 5-5 Diagnostic LED Test Prompts

```
Menu Choice <0> : 8
All leds turn ON? <N>: y
All leds turn OFF? <y>: █
```

Figure 5-9 Diagnostic LED Location



Flash Menu

Select **10. Flash Menu** to display and access selected operations to a specified partition in Flash memory. Figure 5-10 shows an example of this menu. For more information on Flash partitioning, see *"Section 4.6 "Additional Features"*.



NOTE

The Flash Menu options require decimal values only.

Figure 5-10 Flash Menu

```
Menu Choice <0> : 10
1. Show Flash Info
2. Erase Flash Partition
3. Read Flash Partition
4. Write Flash Partition
5. Update Bootloader Partition
6. Update LINUX Partition
0. Exit
Menu Choice (0) : 5
filename : u-boot-octeon_wnpa3850.distro
INFO: Attempting to update the bootloader...
INFO:   Flashing u-boot-octeon_wnpa3850.distro to /dev/mtd0
Jan  1 01:22:01 (none) user.info npaDiag: Attempting to update the bootloader..
INFO:   file u-boot-octeon_wnpa3850.distro has been validated
Jan  1 01:22:01 (none) user.info npaDiag:   Flashing u-boot-octeon_wnpa3850.dis
Jan  1 01:22:01 (none) user.info npaDiag:   file u-boot-octeon_wnpa3850.distro
INFO:   Erased /dev/mtd0
Jan  1 01:22:06 (none) user.info npaDiag:   Erased /dev/mtd0
INFO:   Copied 349920 bytes from u-boot-octeon_wnpa3850.distro to address 0x0000
Jan  1 01:22:08 (none) user.info npaDiag:   Copied 349920 bytes from u-boot-oct
INFO:   Wrote u-boot-octeon_wnpa3850.distro to /dev/mtd0
INFO:   bootloader upgrade was successful
```

Table 5-6 Flash Menu Options

Option	Description
1 Show Flash Info	Displays selected information about the Flash.
2 Erase Flash Partition	Erases a whole Flash partition.
3 Read Flash Partition	Initiates a read to a specified Flash partition.
4 Write Flash Partition	Initiates a write to a specified Flash partition.
5 Update Bootloader Partition	Downloads a new copy of U-Boot bootloader to the specified partition.
6 Update Linux Partition	Updates the Linux partition by downloading a new copy of Linux to the specified partition.
0 Exit	Exits from the Flash Menu and returns to the Main Menu.

Log File Location

Select **11. Log File Location** to display the path to the status and log files as shown in Figure 5-11.

Figure 5-11 Log File Location

```
Menu Choice (11) : 11
Current dependent file location:
  status: /home/root/log/npadiag_8011410_A.log
  logfile: /home/root/log/npastatus_8011410_A
```

Show Core Temperature

Select **12. Show Core Temperature** to display the local and remote temperature settings (in Celsius) as shown in Figure 5-12.

The `local` temperature is the temperature reading of the Local Temperature Sensor. The `remote` temperature is the temperature reading of the Oction internal sensor as read by the Local Temperature Sensor.

Figure 5-12 Show Core Temperature

```
Menu Choice (12) : 12
temperature: local temp 28C, remote 45C
```

Memory Test

Select **13. Mem Test** to run either a DDR3 or PMEM memory test, and to display results as shown in Figure 5-13. The Diagnostic Utility prompts you to select either DDR3 or PMEM:

- When you select DDR (0), `npa_mem_test` named `alloc` must be created *before* booting Linux.
- When you select PMEM (1), no preconditions are required.

After prompting you to select a memory test, the Diagnostic Utility also prompts you to Start (1) or Stop (0) the test.

To automatically run the memory test from the command line, use the `-a` Command Line Parameter, which displays similar to that shown in Figure 5-14.



NOTE

Before booting Linux, create `npa_mem_test` named `alloc` reserved memory space from within U-Boot. For example, `namedalloc npo_mem_test <size> <addr>`. Failure to create the memory space will not allow the DDR test to run.

Figure 5-13 Memory Test Results

```
Menu Choice <0> : 13
Select memory test <DDR3> Mem = 0
Start or Stop? <start = 1, stop = 0>: 1
Starting memory test thread for cpu0
Starting memory test thread for cpu1
Starting memory test thread for cpu2
Starting memory test thread for cpu3
Starting memory test thread for cpu4
Starting memory test thread for cpu5
Starting memory test thread for cpu6
Starting memory test thread for cpu7
Starting memory test thread for cpu8
Starting memory test thread for cpu9
```

Figure 5-14 Memory Test with Command Line Parameter `-a`

```
Starting memory test thread for cpu0
Starting memory test thread for cpu1
Starting memory test thread for cpu2
Starting memory test thread for cpu3
Starting memory test thread for cpu4
Starting memory test thread for cpu5
Starting memory test thread for cpu6
Starting memory test thread for cpu7
Starting memory test thread for cpu8
Starting memory test thread for cpu9

..... *
Stopping memory test thread for cpu0
Stopping memory test thread for cpu1
Stopping memory test thread for cpu2
Stopping memory test thread for cpu3
Stopping memory test thread for cpu4
Stopping memory test thread for cpu5
Stopping memory test thread for cpu6
Stopping memory test thread for cpu7
Stopping memory test thread for cpu8
Stopping memory test thread for cpu9
```

USB Storage Test

Select **15. USB Storage Test** to run a read /write test on the USB when the device is available and mounted. When you select option 15, the diagnostic indicates that it is writing and reading a file to/from the USB. The Main Menu displays when the test completes similar to Figure 5-15.

Figure 5-15 USB Storage Test

```
npa_mem_test namedalloc is not set, skipping DRAM memory tests
*** GE Intelligent Platforms WANic 66XX Diagnostics vx.xx.x NPU: A ***
1. View Test Results          10. Flash Menu
2. EEPROM Menu               11. Log File Locations
3. PCI Configuration Menu    12. Reserved
4. Peek/Poke Menu            13. MEM Test
5. Run Auto Test             14. Reserved
6. Auto Test Configuration Menu 15. USB Storage Test
8. Diagnostic LED test
0. Exit
Menu Choice (0) : 15
Found USB Storage device /dev/sda mounted at /home/root/usb
Writing file to USB storage device.....De
Reading file from USB storage device.....De
*** GE Intelligent Platforms WANic 66XX Diagnostics vx.xx.x NPU: A ***
1. View Test Results          10. Flash Menu
2. EEPROM Menu               11. Log File Locations
3. PCI Configuration Menu    12. Reserved
4. Peek/Poke Menu            13. MEM Test
5. Run Auto Test             14. Reserved
6. Auto Test Configuration Menu 15. USB Storage Test
8. Diagnostic LED test
0. Exit
Menu Choice (15) : █
```

5.3 In-Service Daemon

The In-Service Daemon provides a runtime background process that monitors hardware continuously and reports detected faults. The In-Service Daemon is dependent upon the NPA Driver. The In-Service Daemon logs all errors to the Linux syslog and `/var/log/messages`.

To run the In-Service Daemon, enter the following command:

```
# npaDaemon <option> (see Table 5-7)
```

Table 5-7 In-Service Daemon Options

Option	Description
-h	Displays all command line parameters
-f	Runs in non-daemon mode; runs in foreground
-i <n seconds greater than 10>	Number of seconds to wait before the next service check

When you enter the `cat /var/log/messages` command, a screen similar to Figure 5-16 displays.

Figure 5-16 Error Message Display

```
~ # cat /var/log/messages
Mar 1 00:00:10 (none) daemon.info init: Starting pid 123, console /dev/ttyS0
Mar 1 00:01:34 (none) user.info npaDaemon: Found PHY [0] OUI: 0x456
Mar 1 00:01:34 (none) user.info npaDaemon: Found PHY [1] OUI: 0x456
Mar 1 00:01:34 (none) user.info npaDaemon: Found PHY [2] OUI: 0x456
Mar 1 00:01:34 (none) user.info npaDaemon: Found PHY [3] OUI: 0x456
Mar 1 00:01:34 (none) user.info npaDaemon: Flash Primary device size 128MB four
Mar 1 00:01:34 (none) user.info npaDaemon: FPGA version is 0.12
Mar 1 00:01:34 (none) user.info npaDaemon: npaDaemon V1.04 running
Mar 1 00:01:34 (none) user.info npaDaemon: Found PHY [3] OUI: 0x456
Mar 1 00:01:34 (none) user.info npaDaemon: Found PHY [0] OUI: 0x456
Mar 1 00:01:34 (none) user.info npaDaemon: Found PHY [1] OUI: 0x456
Mar 1 00:01:34 (none) user.info npaDaemon: Found PHY [2] OUI: 0x456
Mar 1 00:01:34 (none) user.info npaDaemon: Found PHY [3] OUI: 0x456
Mar 1 00:01:34 (none) user.info npaDaemon: Flash Primary device size 128MB four
Mar 1 00:01:34 (none) user.info npaDaemon: WNPA56xx FPGA check
Mar 1 00:01:34 (none) user.info npaDaemon: WNPA56xx Local PHY Status
Mar 1 00:01:34 (none) user.info npaDaemon: WNPA56xx Flash Device Status
Mar 1 00:01:34 (none) user.info npaDaemon: Test sequence count is [1]
~ # █
```


6 • Specification

This chapter contains product specifications and regulatory compliance notices.

6.1 Hardware Specifications

Table 6-1 lists the hardware specifications for the released configurations of the WANic-66512. Table 6-2 lists the regulatory compliance notices for this product.

Table 6-1 Hardware Specifications

Description	Model WPC6D74010A (1.3 GHz Processor)	Model WPC6D84010B (1.5 GHz Processor)
Form Factor	PCI-Express-compliant, Standard-height half-length (4.2 inches x 6.6 inches) (10.668 centimeters x 16.764 centimeters)	
Slot Type	Requires Four Lane PCI Express slot	
Bus Type	PCI Express 2.0 compliant	
PCI Express to Edge Connector	4 lanes	
I/O Configuration	Two front panel optical or copper SFPs,	
Core Processors	10	10
Processor Core Speed	1.3 GHz	1.5 GHz
Flash ROM	128 MB	
Flash Disk eUSB	2 GB	
Persistent Memory	32 MB	
DDR3 SDRAM	8 GB total (4 GB per slot)	
EEPROM	64 KB	
IEEE Time Synchronization and Time Stamp Unit	Supported	
HFA/DFA Pattern Search	512 MB of DDR3 SDRAM	
Power Requirements		
Watts (typical maximum)	Less than 53.2 W	Less than 61.3 W
Mean Time Between Failure (Telcordia Method 1 Case 3 @ Ground Benign, Controlled)		
Main Card /no SFP+	217,116 hours @ 40°C	217,116 hours @ 40°C
Environmental Requirements		
Ambient Operating Temperature	0° to +55°C (+32° to +131°F)	0° to +50°C (+32° to +122°F)
Storage Temperature	-40° to +85°C (-40° to +185° F)	
Relative Humidity	5% to 95% (non-condensing)	
Shock	5G each axis	
Vibration	5-500Hz, 1G axis	
Altitude	4,572/15000 ft.	

6.2 Regulatory Compliance Notice

This section identifies the regulatory compliance notices for the WANic-66512.

WANic-66512 Model WPC6D74010A (1.3 GHz Processor) is designed for Network Equipment Building Systems (NEBS) compliance.

Table 6-2 Regulatory Compliance

Compliance	Regulatory Notice
RoHS	6/6
EMC Immunity	Europe — EN55024:1998/A1:2001/A2:2003 EM300 386 v1.4.1
Safety	Canada – CSA22.2 NO 60950-1, Second Edition Europe – EN60950-1, Second Edition USA – UL60950-1, Second Edition
Flammability	UL94V0
Emissions Class A	Australia – AS/NZS CISPR 22:2006, Class A ITE Canada – ICES-003 Issue 4, Class A Japan – VCCI, Class A ITE USA – FCC 47 CFR Part 15, Class A
CE Notice	Europe – EN55022:2006/A1:2007, Class A ITE
Harmonic Current Emissions	Europe - EN 61000-3-2:2006 Europe - EN 61000-3-3:1995/A1:2001/A2:2005

6.2.1 Emission Notices

This equipment complies with the following international and North American emission requirements:

- **Australia and New Zealand** – AS/NZS 3548/CISPR 22 Class A ITE

- **Canada** – ICES-003

This Class A digital apparatus complies with Canadian ICES-003.
(Cet appareil numérique de la class A est conforme a la norme NMB-003 du Canada.)

- **Japan** – VCCI Class A ITE

This is a Class A product based on the standard of the Voluntary Control Council for Interference from Information Technology Equipment (VCCI). If this is used near a radio or television receiver in a domestic environment, it may cause radio interference. Install and use the equipment according to the instruction manual.

- **USA** – FCC Part 15 Class A



NOTE

The hardware has been tested and found to comply with the limits for a Class A digital device pursuant to Part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction in this documentation, may cause harmful interference to radio communications. Operation of the equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense. Changes or modification not expressly approved by the manufacture could void the user's FCC granted authority to operate the equipment.

A • GE Intelligent Platforms, Inc. Software License Description

This document describes the licensing of the individual software components included on this product.

The following components are included as firmware provided in Flash memory.

A.1 U-Boot

The U-Boot bootloader firmware is covered by the GNU General Public License (GPL), version 2. Please see the GNU GPL v2, Part Number: 21-LIC-GPLV2-0 for more information regarding the terms of this license.

A.2 Linux Image

The Linux image (kernel and embedded file system) is covered by the GNU GPL v2. Please see the GNU GPL v2, Part Number: 21-LIC-GPLV2-01 for more information regarding the terms of this license.

B • GNU General Public License V2

The following information is exactly as provided by the Free Software Foundation, Inc.

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA.

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps:

1. copyright the software, and
2. offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions For Copying, Distribution and Modification

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:
 - a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
 - b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
 - c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
 - a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
 - b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
 - c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.
6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and an idea of what it does.  
Copyright (C) 19yy name of author  
This program is free software; you can redistribute it and/or  
modify it under the terms of the GNU General Public License as  
published by the Free Software Foundation; either version 2 of the  
License, or (at your option) any later version.
```

```
This program is distributed in the hope that it will be useful, but  
WITHOUT ANY WARRANTY; without even the implied warranty of  
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU  
General Public License for more details.
```

```
You should have received a copy of the GNU General Public License  
along with this program; if not, write to the Free Software  
Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-  
1307, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) 19yy name of author  
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type  
'show w'. This is free software, and you are welcome to  
redistribute it under certain conditions; type 'show c' for  
details.
```

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the  
program 'Gnomovision' (which makes passes at compilers) written by  
James Hacker.
```

```
signature of Ty Coon, 1 April 1989  
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

FSF & GNU inquiries & questions to *gnu@prep.ai.mit.edu*. Other ways to contact the FSF.

Comments on these web pages to *webmasters@www.gnu.ai.mit.edu*, send other questions to *gnu@prep.ai.mit.edu*.

Copyright notice above.

Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111, USA

Updated: 22 Jul 1997 tower

This glossary contains a listing of terms, acronyms, and definitions.

10GbE	10 Gigabit Ethernet connectivity, which provides transmission speeds up to 10 billion bits per second. 10GbE uses the IEEE 802ae protocol.
DDR3	Double Data Rate Type 3. A type of synchronous DRAM Technology that permits the transfer of two words of data per clock period.
DFA	Deterministic Finite Automata.
DIMM	Dual in-line Memory Module.
DRAM	Dynamic Random Access Memory.
EEPROM	Electrically Erasable Programmable Read Only Memory. A type of non-volatile memory that permits its content to be selectively modified on a location-by-location basis. Flash memory is a special category of EEPROM, which can be reprogrammed on a sector-by-sector basis or in its entirety only after the prior content of those sectors is erased.
ECC	Error Correction Code.
E-Keying	Electronic-keying defines the process by which the carrier blade determines the compatibility of the control and fabric interfaces on a module with the carrier interconnects.
ESD	ElectroStatic Discharge.
Ethernet	Ethernet is a local area network technology specified in the IEEE 802.3 standard.
Flash Memory	Sometimes called Flash RAM, Flash memory is a type of constantly-powered nonvolatile memory that can be erased and reprogrammed in units of memory called blocks.
GB	GigaByte.
Gigabits per second (Gbps)	Gigabits per second is one billion bits per second.
Gigabit Ethernet (GbE)	Gigabit Ethernet provides higher level of backbone support at 1000 megabits per second (1 gigabits or 1 billion bits per second).
GND	Ground provides connections to circuit ground at both ends. Ground ensures that the transmit signal levels stay within the common mode input range of receivers.
GMII	Gigabit Media Independent Interface provides a simple interconnection between MAC and PHY device that is independent of the physical media types.
GPIO	General Purpose Input and Output.
HFA	Hyper-Finite Automata.
IC	Integrated Circuit.

I2C	Inter-Integrated Circuit (I2C). Two-wire interface commonly used to connect low-speed peripherals in an embedded system.
IEEE	Institute of Electrical and Electronic Engineers, Inc standards organization.
IEEE 802.3	A local area network protocol suite commonly known as Ethernet.
I/O	Input/Output.
IP	Internet Protocol.
JTAG	Joint Action Test Group
KB	KiloByte.
LED	Light Emitting Diode.
LFM	Linear Feet per Minute.
LOS	Loss of Signal.
MAC	Media Access Control.
MB	MegaByte.
Mbps	Megabits per second.
MDI	Medium Dependent Interface.
MDIO	Management Data Input/Output
MPSC	Multi-Protocol Serial Controller.
MPX	Multi-Processor Extension.
ms	A millisecond (from milli- and second) is a thousandth ($1/1,000$) of a second.
MTBF	Mean Time Between Failure.
Multi-Core Processor	A Multi-Core Processor is an integrated circuit to which two or more processors have been attached for enhanced performance, reduced power consumption, and more efficient simultaneous processing of multiple tasks.
NFA	Nondeterministic Finite Automata.
NMI	Non-Maskable Interrupt.
NVRAM	Non-Volatile Random Access Memory.
OEM	Original Equipment Manufacturers.
PCI	Peripheral Component Interconnect.
PCIe	PCI Express.
PHY	A generic electronics term referring to a special electronic integrated circuit or functional block of a circuit that provides PHYsical access to a digital connection cable. Also know as a PHYsical layer device.
PICMG	PCI Industrial Computer Manufacturers Group.

Ping	A basic Internet program that allows a user to verify that a particular IP address exists and can accept requests.
PPM	Part per million. Crystal Clock accuracy is defined in terms of parts per million and it gives a convenient way of comparing accuracies of different crystal specifications
QLM	Quad-Lane Modules.
RGMII	Reduced GMII (See GMII.)
RoHS	The European Restriction of Hazardous Substance in electrical and electronic equipment. sold or used in the European Union after July 1, 2006. These substances are lead, mercury, cadmium, hexavalent chromium, polybrominated biphenyls, and polybrominated diphenyl ethers.
ROM	Read Only Memory.
SDRAM	Synchronous Dynamic Random Access Memory.
SerDes	Serializer Deserializer.
SFP+	Small Form-factor Pluggable Plus.
SGMII	Serial Gigabit Media Independent Interface.
ShMC	Shelf Management Controller.
SMBus	System Management Bus. Similar to I2C.
SyncE	Synchronous Ethernet.
TCK	Test Clock.
TDI	Test Data In.
TDO	Test Data Out.
TEM	Telecommunications Equipment Manufacturers.
TMS	Test Mode Select.
Tuple	An ordered set of values.
TWSI	Two-Wire Serial interface.
UART	Universal Asynchronous Receiver Transmitter.
USB	Universal Serial Bus.
Vaux	Vaux power is designed to provide limited power to the PCI bus devices in the event the primary power supply is off.
XAUI	An interface employed by 10Gigabit Ethernet, where X stands for the Roman numeral ten and AUI is derived from the Ethernet Attachment Unit Interface.

Numerics

20-Pin Header	
10-pin breakout	61
14-pin breakout	61

A

Active script tuple	119
Application executable entries	106
Auto test	159

B

Board ID and DIP Switch Register	71
Boot	
bus	39
firmware	89
Flash	97

C

Cable	
IDE	61
serial debug adapter	61
Cavium SDK	
Embedded File System,	92
OCTEON Ethernet Driver	91
Simple Executive (Exec) Library	94
Character device	153
Checksum tuple	110
CIU_PP_POKE _n Register	46
CIU_WDOG _n Register	46
Command Line Parameters	152
Command script	105
Connector	
PCI Express	57
SFP+	25
CSUMTESTI tuple	108

D

DB-9 connector	61
DDR2 SDRAM	53
DHCP	105
Diagnostic LEDs	72, 161

Diagnostic Utility

Auto Test Configuration Menu	159
cable	159
Command Line Parameters	152
Diagnostic LED Test	161
EEPROM menu	157
Flash Menu	162
Log File Location	163
Main Menu	155
memory test	165
run auto test	158
Show Core Temperature	164
view test results	156
DIMMs	
location	53
POST test	96
reset	54, 75
Serial Presence Detect	54
Thermal Event Register	88
TWSI interface	44
DIP Switch	64
illustration	64
register	64
Dust plugs	17, 27

E

EEPROM	55, 107
mapping	107
programming	157
tuple API	128
tuple PMEMTESTI	114
tuples	108
EJTAG Header	63
Emission compliance	170
Environment	
requirements	169
variables	103
Environmental Requirements	19
Examples	
intercepts	148
Linux application	145
LSP	145
Simple Exec application	147
External JTAG (EJTAG)	63

	F			I
Firmware		89	I2C Address High Register	81
Flash		54	I2C Address Low Register	81
accessing		131	I2C Control Register	80
base address		130	I2C Data Register	82
device identification		129	I2C registers	78
displaying partitions		131	IDE cable	61
functions		129	Inserting an SFP module	28
mapping		130	Installation	
partitioning		131	precautions	27
programming		162	WANic card	30
running the file system		132	Interrupt	
FPGA		49	Control Register	73
base address		67	generation	73
clocking		50	Status Register	74
power control		51	ioctl	
register mapping		50	commands	137
registers		67	examples	136
synchronous Ethernet clock		48	IR Subsystem	58
Front panel			IR3541 Multiphase Controller	58
description		24	IR3550 synchronous buck gate driver	58
SFP connectors		24		
	G			J
Gigabit Ethernet PHY test tuple		113	J4 Header	21
GNU General Public License		173	J5 Header	60
GPIO			J6 Header	62
interface		44	JTAG scan chain	62
pins		44		
	H			L
Handling precautions		16	LABI tuple	108, 111
Hardware			LED Control Register	72
DIMM slots		53	LEDs	
FPGA		49	description	24
headers		60	Link and activity	26
initialization		54	Power	26
memory		53	Level 2 cache	53
Multi-core Processor		37	Linux Support Package (LSP)	127
overview		35	examples	145
PMEM		54		
serial EEPROM		55		
USB		55		
XL30152		48		
Header				M
EJTAG		63	Memory	53
J4		21	DDR2 SDRAM	53
J5		61	EEPROM	36, 55
J6		62	Flash	54
HFA/DFA		39, 54	HFA/DFA	54
			MEMTEST tuple	108
			Minimum volumetric air flow	20

Module		Registers	
installation	30	I2C Address High	81
removal	32	I2C Address Low	81
mtd_debug	131, 132	I2C Control	80
Multi-Core Processor	37	I2C Data	82
configuration	37	Interrupt Status	74
fan controller	59	Payload Reset	75
GPIO interface	43	Transmit Control	76
Level 2 cache	53	Related specifications and documentation	11
power supply	38	Removing	
TWSI interface	41	dust plugs	28
UARTs	45	SFP+ module	29
		WANic card	32
	N	Reprogramming	
Notation convention	13	boot loader	97
NPA Driver	128, 134	Linux Kernel in Flash	132
		U-Boot	132
		RLDRAM tuple	116
	O		
OCTEON Ethernet driver	91		S
On-Board Thermal Protection	52	S1 DIP Switch	62, 64
Optical/fiber transceiver	25	SCRIPT tuple	108
		SCRIPTACTIVE tuple	108
	P	Secondary File Information (SFI)	104
Payload Reset Register	75	SerDes tuple	117
PCI Express		SERDESTEST1 tuple	108
interface	40	Serial Debug Adapter (SDA) cable	61
connector	57	Serial Presence Detect	54
Persistent memory	54	SFI tuple	108, 121
PFI tuple	108, 115	SFP module	27
PHY device	56	holding	29
PHYTESTI tuple	108	inserting	28
POST error codes	96	removing	29
Power		storing	29
distribution	57	unlocking	29
edge fingers	57	Simple Exec Library	94
IR Subsystem	58	SIPI tuple	108
limits	21	Software	
supplies	21	components	89
supply control	51	Developer's Kit	33
Primary File Information (PFI)	104	GNU General Public License	173
Primitives, TWSI	128	LSP	151
Proc file	135	operating systems	34
		Source IP Information (SIPI)	104
	Q	Static-protective packaging	16
QT2025(See PHY device)		SyncE	48
Quad-Lane Modules (QLMs)	40	reset	75
	R		
Reading data	79		
Reference Clock	47		

	T		V
Temperature		Voltage	22
monitoring from the host	59		
sensor	59		
thresholds	59	W	
Thermal Protection	52	Watchdog	45
Transmit Control Register	76	Watchdog Timer	45
Tuple	95	Writing data	79
CSUMEMTESTI	110		
CSUMTESTI	110	Z	
deleting	109	ZL30152 SyncE Device(See SyncE)	
displaying an address	109		
EEPROM	108		
enumeration	125		
format	108		
LABI	111		
PFI	115		
PHYTESTI	113		
RLDRAMTI	116		
SCRIPT	118		
SCRIPTACTIVE	119		
SERDESTESTI	117		
SFI	121		
SIPI	120		
UARTCONI	122		
UARTMMCI	123		
uboot_normal_info	124		
TWSI			
interface	41		
primitives	128		
	U		
UA			
composition	94		
loading	94		
loading into DDR3 SRAM	106		
UARTCONI tuple	108		
UARTs	45		
tuple	122		
U-Boot	95		
building	97		
commands	98		
nitialization	95		
POST error codes	96		
POST tests	96		
reprogramming	97, 132		
tuple	95		
Unlocking the SFP module	29		
Unpacking	23		
USB Storage Test	166		
User Application (See UA)			

© 2012 GE Intelligent Platforms Embedded Systems, Inc. All rights reserved.

An asterisk (*) indicates a trademark of GE Intelligent Platforms, Inc. and/or its affiliates. All other trademarks are the property of their respective owners.

Confidential Information - This document contains Confidential/Proprietary Information of GE Intelligent Platforms, Inc. and/or its suppliers or vendors. Distribution or reproduction prohibited without permission.

THIS DOCUMENT AND ITS CONTENTS ARE PROVIDED "AS IS", WITH NO REPRESENTATIONS OR WARRANTIES OF ANY KIND, WHETHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO WARRANTIES OF DESIGN, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. ALL OTHER LIABILITY ARISING FROM RELIANCE UPON ANY INFORMATION CONTAINED HEREIN IS EXPRESSLY DISCLAIMED.

GE Intelligent Platforms Information Centers

Americas:

1 800 322 3616 or 1 256 880 0444

Asia Pacific:

86 10 6561 1561

Europe, Middle East and Africa:

Germany +49 821 5034-0

UK +44 1327 359444

Additional Resources

For more information, please visit the GE Intelligent Platforms Embedded Systems web site at:

www.ge-ip.com

